convertToXml

```
protected Element convertToXml(Message msg, Document doc)
```

return an XML Message element from this message object

Throws:
MessagingException

generateData

```
protected void generateData()
```

Populate the cached data for outgoing messages.

Throws:
MessagingException

getDTD

```
public DTD getDTD()
```

return a DTD describing the XML that can be retreived using the XML Stream. Note: Not Implemented yet; just returns null.

getFolderInfoElement

```
protected Element getFolderInfoElement(Folder folder, Document doc)
```

Return an Element with information about the folder

getHost

```
public String getHost()
```

tgGet

getMailbox

```
public String getMailbox()
```

tgGet

getPassword

```
public String getPassword()
```

tgGet

getSaveables

```
public Object[] getSaveables()
```

Return an array of all objects that should be persistent. primitive datatypes must be converted to their Object equivalent. These objects will come back in the same order when setSaveables is called.

getUserAcct

```
public String getUserAcct()
```

tgGet

getXmlElement

```
protected Element getXmlElement(String key, String value, Document doc)
```

return an element with this name and value

getXMLStream

```
public InputStream getXMLStream()
```

Return an input stream by which an XML document can be read. The XML Stream does NOT included DTD info.

Throws:
NewEmailQEException

openMessageStore

```
protected void openMessageStore()
```

Open the message store. Called from generateData()

Throws:
NoSuchProviderException
MessagingException

requery

```
protected void requery()
```

Refresh the view of the external data source. By default, if the requery() call is made in an interval < the queryInterval from the lastCheckDate, then nothing will happen.

setHost

```
public void setHost(String host)
```

tgSet

setMailbox

```
public void setMailbox(String mailbox)
```

tgSet

setPassword

```
public void setPassword(String password)
```

tgSet

setSaveables

```
public void setSaveables(Object[] oa)
```

receive an array of objects that have been saved. the order will correspond to that specified when getSaveables was called. Users should handle Null values in the array.

setUserAcct

```
public void setUserAcct(String userAcct)
```

tgSet

Class *tcs.ain.infoserver.NewEmailQueryEngine.XMLInputStream*

```
class XMLInputStream
```

Extends:
java.io.InputStream

This handles access to the XML data through an input stream. Each object to request the inputstream gets one of these, which handles sending the data to the client from the bytearray data. Can be used by children at their discretion (e.g., if spooling cached data)

Attributes

lastPos

Operations

read()

Attribute Detail

lastPos

```
int lastPos = -1
```

Operation Detail

read

```
public int read()
```

Throws:
IOException

Class *tcs.ain.infoserver.NotifyChangeOnlyDestination*

```
public class NotifyChangeOnlyDestination
```

Extends:
tcs.ain.infoserver.UserDeliveryDestination

Attributes

debug
lastChecksum

Constructors

NotifyChangeOnlyDestination()

Operations

cleanUp()
getChecksum(java.lang.String)
notifyDestinations(tcs.ain.infoserver.DataWorker, tcs.ain.infoserver.IDataSource)
processData(tcs.ain.infoserver.DataEvent)

Attribute Detail

debug

```
protected boolean debug
```

lastChecksum

```
protected long lastChecksum
```

Constructor Detail

NotifyChangeOnlyDestination

```
public NotifyChangeOnlyDestination()
```

Operation Detail

cleanUp

```
public void cleanUp()
```

getChecksum

```
protected long getChecksum(java.lang.String)
```

notifyDestinations

```
protected void notifyDestinations(tcs.ain.infoserver.DataWorker, tcs.ain.infoserver.IDataSource)
```

processData

```
public void processData(tcs.ain.infoserver.DataEvent)
```

Class *tcs.ain.infoserver.PageParserQEException*

```
public class PageParserQEException
```

Extends:

tcs.ain.infoserver.InfoServerException

Constructors

PageParserQEException(String)

Constructor Detail

PageParserQEException

```
public PageParserQEException(String s)
```

Class *tcs.ain.infoserver.PageParserQueryEngine*

```
public class PageParserQueryEngine
```

Extends:

tcs.ain.infoserver.QueryEngine

This query engine uses the TCS page parser to examine an html/xml document. The page parser provides access to the components of a page according to the underlying document structure. This engine allows the user to specify the portions of the page that are to be retrieved. Document structure is according to common HTML tag elements if no targetElements are defined, then all are returned

Attributes

ba
filename
startRef
targetedElements

Constructors

PageParserQueryEngine()

Operations

addTargetedElement(String, String)
generateData()
getDTD()
getElementKeyWithName(String)
getFilename()
getSaveables()
getStartRef()
getTargetedElementKeys()
getTargetedElements()
getXMLAlias(String)
getXMLStream()
includeData(String)
removeTargetedElement(String)
requery()
setFilename(String)
setSaveables(Object[])
setStartRef(String)
setTargetedElements(Hashtable)

Attribute Detail

ba

```
protected ByteArrayOutputStream ba
```

This contains data that can be returned in the InputStream.

transient

filename

```
protected String filename
```

The name of the file to process; name can be local filename or http address

startRef

```
protected String startRef = new String()
```

Defines the some text to look for in the page to trigger when we started parsing it; if empty then we start parsing at the beginning

targetedElements

```
protected Hashtable targetedElements = new Hashtable()
```

Contains the document structure value and an (optional) label to use in the XML structure we transmit format: page element : XML label

Constructor Detail

PageParserQueryEngine

```
public PageParserQueryEngine()
```

Construct with the specified page name to parse; can be local filename or http address.

Operation Detail

addTargetedElement

```
public void addTargetedElement(String elementName, String xmlLabel)
```

Assign a targeted element; provide the actual element name along with an alias to be provided

generateData

```
protected void generateData()
```

Populate the ba array with XML data from the page parser If no targetElements are defined, we return all document data If no XML are explicitly defined for the elements, then the XML label is the document element label

Throws:
IOException

getDTD

```
public DTD getDTD()
```

return a DTD describing the XML that can be retreived using the XML Stream. Note: Not Implemented yet; just returns null.

getElementKeyWithName

```
protected String getElementKeyWithName(String name)
```

Get the targetedElements key object having this name null if not found

getFilename

```
public String getFilename()
```

return a DTD describing the XML that can be retreived using the XML Stream. Note: Not Implemented yet; just returns null.

getSaveables

```
public Object[] getSaveables()
```

Return an array of all objects that should be persistent. primitive datatypes must be converted to their Object equivalent. These objects will come back in the same order when setSaveables is called.

getStartRef

```
public String getStartRef()
```

Return the assigned text that triggers when the pageParser will start parsing the page.

getTargetedElementKeys

```
public Enumeration getTargetedElementKeys()
```

Return the Enumeration of targeted Elements that have been defined for inclusion in the XML data

getTargetedElements

```
public Hashtable getTargetedElements()
```

getXMLAlias

```
public String getXMLAlias(String element)
```

Return the XML label defined for the specified targeted element Null if none defined

getXMLStream

```
public InputStream getXMLStream()
```

Return an input stream by which an XML document can be read. The XML Stream does NOT included DTD info.

Throws:

PageParserQEException

includeData

```
protected boolean includeData(String keyStr)
```

Return whether we should include data in the XML document for the documentElement of the targeted page If no targetedElements have been defined, then always true. Otherwise, only if a targetedElement has been previously defined for the document element

removeTargetedElement

```
public void removeTargetedElement(String elementName)
```

remove the element identifier from the list of elements to target in the XML document

requery

```
protected void requery()
```

Refresh the view of the external data source. By default, if the requery() call is made in an interval < the queryInterval from the lastCheckDate, then nothing will happen.

Throws:

IOException

setFilename

```
public void setFilename(String filename)
```

return a DTD describing the XML that can be retreived using the XML Stream. Note: Not Implemented yet; just returns null.

setSaveables

```
public void setSaveables(Object[] oa)
```

receive an array of objects that have been saved. the order will correspond to that specified when getSaveables was called. Users should handle Null values in the array.

setStartRef

```
public void setStartRef(String ref)
```

Assign the text that triggers when the pageParser will start parsing the page. If not defined, the page parser will start from the beginning. Otherwise it will only start parsing when it finds the text in the displayed text or in the param string of a tag

setTargetedElements

```
public void setTargetedElements(Hashtable targetedElements)
```

Class *tcs.ain.infoserver.PageParserQueryEngine.XMLInputStream*

```
class XMLInputStream
```

Extends:

java.io.InputStream

This handles access to the XML data through an input stream. Each object to request the inputstream gets one of these, which handles sending the data to the client from the bytearray data. Can be used by children at their discretion (e.g., if spooling cached data)

Attributes

lastPos

Operations

read()

Attribute Detail

lastPos

```
int lastPos = -1
```

Operation Detail

read

```
public int read()
```

Throws:

IOException

Class *tcs.ain.infoserver.pqa.LCDemoPublicDS*

```
public class LCDemoPublicDS
```

Tests the IMAP Query Engine; query every one minute

Constructors

LCDemoPublicDS()

Operations

main(String[])

Constructor Detail

LCDemoPublicDS

```
public LCDemoPublicDS()
```

Throws:

Exception

Operation Detail

main

```
public static void main(String[] args)
```

Class *tcs.ain.infoserver.pqa.PQAEBook*

```
public class PQAEBook
```

Extends:

tcs.ain.infoserver.pqa.PQAPage

Launch a DataWorker for a given Agent, and return results

Attributes

AGENT_ID
ATTRVAL
HtmlPage
LCRESULTS1_PATH
TAG
workerId
WORKER_ID

Constructors

PQAEBook(HttpServletRequest, HttpServletResponse, String)

Operations

getBody()
getForm(String)
GetPageId()
getTagTable(String)

Attribute Detail

AGENT_ID

```
protected static final String AGENT_ID = "rsmith"
```

Agent ID

ATTRVAL

```
protected static final String ATTRVAL = "SF_ATTRSTR"
```

Attribute Value parameter name constant

HtmlPage

```
protected static StringBuffer HtmlPage = null
```

Stores Html Page associated with a form producing class

LCRESULTS1_PATH

```
public static final String LCRESULTS1_PATH = QNetPQAResource.getString("LCResults1")
```

Results Page Path

TAG

```
protected static final String TAG = "SF_TAG"
```

Tag parameter name constant

workerId

```
protected String workerId = ""
```

Stores Html Page associated with a form producing class

WORKER_ID

```
protected static final String WORKER_ID = "LC Operation Manual"
```

WorkerID

Constructor Detail

PQAEBook

```
public PQAEBook(HttpServletRequest request, HttpServletResponse response, String userName)
```

Constructor calls super() and sets the title for this page

Operation Detail

getBody

```
public String getBody()
```

get Login Body. Since the initial login form is stored on the pqa, this should only be called to retrieve an error login

getForm

```
protected String getForm(String results)
```

Create an html form for selecting or removing existing Data Workers. This is the initial form for the Query Net Wizard

GetPageId

```
public static int GetPageId()
```

getTagTable

```
protected Hashtable getTagTable(String results)
```

Create a table of tags and their values to be used in replacing tags found in static html pages. Used to add dynamic content to these pages.

Class *tcs.ain.infoserver.pqa.PQALauncher*

```
public class PQALauncher
```

Extends:

tcs.ain.infoserver.pqa.PQAPage

Launch a DataWorker for a given Agent, and return results

Attributes

HtmlPage
workerId

Constructors

PQALauncher(HttpServletRequest, HttpServletResponse, String)

Operations

getBody()
getForm(String)
GetPageId()
getTagTable(String)

Attribute Detail

HtmlPage

```
protected static StringBuffer HtmlPage = null
```

Stores Html Page associated with a form producing class

workerId

```
protected String workerId = ""
```

Stores Html Page associated with a form producing class

Constructor Detail

PQALauncher

```
public PQALauncher(HttpServletRequest request, HttpServletResponse response, String userName)
```

Constructor calls super() and sets the title for this page

Operation Detail

getBody

```
public String getBody()
```

get Login Body. Since the initial login form is stored on the pqa, this should only be called to retrieve an error login

getForm

```
protected String getForm(String results)
```

Create an html form for selecting or removing existing Data Workers. This is the initial form for the Query Net Wizard

GetPageId

```
public static int GetPageId()
```

getTagTable

```
protected Hashtable getTagTable(String results)
```

Create a table of tags and their values to be used in replacing tags found in static html pages. Used to add dynamic content to these pages.

Class *tcs.ain.infoserver.pqa.PQALogin*

```
public class PQALogin
```

Extends:

tcs.ain.infoserver.pqa.PQAPage

USER LOGIN PAGE. This prompts user for username/pw and tries and to find (or create) a UserAgent object for them. The UserAgent object is inserted into the user's Session under the name 'userAgent', for retention between screens This will eventually validate against and reference the SMS Internet Suite UserInfo database. Note: New agents are created for authenticated users, and their WebProv values will be assigned to the agent for name, email, min We will need to keep the agent info up-to-date somehow.

Attributes

HtmlPage
InvalidLoginMsg

Constructors

PQALogin(HttpServletRequest, HttpServletResponse)

Operations

getBody()
getForm()
GetPageId()
getTagTable()
IsValidLogin(ResultSet, String)
ProcessLogin(HttpServletRequest)
QueryUserInfoDB(String)

Attribute Detail

HtmlPage

```
protected static StringBuffer HtmlPage = null
```

Stores Html Page associated with a form producing class

InvalidLoginMsg

```
protected static final String InvalidLoginMsg = QNetPQAResource.getString("InvalidLoginMsg")
```

Stores Html Page associated with a form producing class

Constructor Detail

PQALogin

```
public PQALogin(HttpServletRequest request, HttpServletResponse response)
```

Constructor calls super() and sets the title for this page

Operation Detail

getBody

```
public String getBody()
```

get Login Body. Since the initial login form is stored on the pqa, this should only be called to retrieve an error login

getForm

```
protected String getForm()
```

Create an html form for selecting or removing existing Data Workers. This is the initial form for the Query Net Wizard

GetPageId

```
public static int GetPageId()
```

getTagTable

```
protected Hashtable getTagTable()
```

Create a table of tags and their values to be used in replacing tags found in static html pages. Used to add dynamic content to these pages.

IsValidLogin

```
protected static boolean IsValidLogin(ResultSet r, String password)
```

Authenticate the user.

ProcessLogin

```
public static String ProcessLogin(HttpServletRequest req)
```

Validate a login, return username if valid log in, null if invalid. REMEMBER: users can log in with Alias or PIN so be sure

QueryUserInfoDB

```
protected static ResultSet QueryUserInfoDB(String username)
```

Query the UserProvisioning database for the login name

Class *tcs.ain.infoserver.pqa.PQAMenu*

```
public class PQAMenu
```

Extends:

tcs.ain.infoserver.pqa.PQAPage

Return a Menu listing of all DataWorkers belonging to a given UserAgent.

Attributes

HtmlPage
NoWorkersPage

Constructors

PQAMenu(HttpServletRequest, HttpServletResponse, String)

Operations

getBody()
getForm(Vector)
GetPageId()
getSelectBoxOptions(Vector)
getTagTable(Vector)

Attribute Detail

HtmlPage

```
protected static StringBuffer HtmlPage = null
```

Stores Html Page associated with a form producing class

NoWorkersPage

```
protected static StringBuffer NoWorkersPage = null
```

Stores Html Page associated with a form producing class

Constructor Detail

PQAMenu

```
public PQAMenu(HttpServletRequest request, HttpServletResponse response, String userName)
```

Constructor calls super()

Operation Detail

getBody

```
public String getBody()
```

get Menu Body. List all DataWorkers associated with a given agent

getForm

```
protected String getForm(Vector workerList)
```

Create an html form for selecting or removing existing Data Workers. This is the initial form for the Query Net Wizard

GetPageId

```
public static int GetPageId()
```

getSelectBoxOptions

```
protected String getSelectBoxOptions(Vector workerList)
```

Create a select box for the given parameter and select a specified value. *

getTagTable

```
protected Hashtable getTagTable(Vector workerList)
```

Create a table of tags and their values to be used in replacing tags found in static html pages. Used to add dynamic content to these pages.

Class *tcs.ain.infoserver.pqa.PQAPage*

```
public abstract class PQAPage
```

Extends:
tcs.web.WebPage

Implements:
tcs.ain.infoserver.pqa.PQAConst

Launch a DataWorker for a given Agent, and return results

Attributes
userName

Constructors
PQAPage(HttpServletRequest, HttpServletResponse)
PQAPage(HttpServletRequest, HttpServletResponse, String)

Operations
getHtmlPage(String)
replaceTags(String, Hashtable)

Attribute Detail

userName
```
protected String userName
```

Constructor Detail

PQAPage
```
public PQAPage(HttpServletRequest request, HttpServletResponse response)
```
Constructor calls super() and sets the title for this page

PQAPage
```
public PQAPage(HttpServletRequest request, HttpServletResponse response, String userName)
```
Constructor calls super() and sets the title for this page

Operation Detail

getHtmlPage
```
public StringBuffer getHtmlPage(String path)
```
Read in specified Html Page and store to static HtmlPage StringBuffer

Throws:
Exception

replaceTags
```
public String replaceTags(String content, Hashtable ht)
```
Read in specified Html Page and store content to StringBuffer

Class *tcs.ain.infoserver.pqa.PQAPagePrinter*

```
public class PQAPagePrinter
```

Extends:
tcs.web.WebPagePrinter

Base class for web page printing operations. Provides methods to print the header, banner, etc. Subclasses can provide distinct color / font

/ banner schemes for individual projects or pages

Constructors

PQAPagePrinter(WebPage)

Operations

print()

Constructor Detail

PQAPagePrinter

```
public PQAPagePrinter(WebPage page)
```

Construct the page printer object. This assings the owner and gets the output stream

Operation Detail

print

```
public void print()
```

public print method; what client calls to start the printing process. This calls the basic components of the page (header, banner, body, footer)

Class *tcs.ain.infoserver.pqa.PQAServletMain*

```
public class PQAServletMain
```

Extends:

tcs.web.WebMain

Implements:

tcs.ain.infoserver.pqa.PQAConst

This is the Main page for the QueryNet PQA Servlet. This loads pages according to the received Section variable

Attributes

cursorCount
cursorLimit
infoServerAccess
jdbcConn
MainApp

Constructors

PQAServletMain()

Operations

cursorAvailable()
getDbConnection()
getRMIConnection()
getWebPagePrinter(WebPage)
loadSection(HttpServletRequest, HttpServletResponse)
makeDbConnection()
makeRMIConnection()
releaseCursor()
showDBError(HttpServletResponse)
showRMIError(HttpServletResponse)
validateDbConnection()
validateRMIConnection()

Attribute Detail

cursorCount

```
protected static int cursorCount = 0
```

The database cursors currently open by repository.

cursorLimit

```
protected static final int cursorLimit = QNetPQAResource.getInt("DbCursorLimit")
```

The database cursor limit for the Repository database connection.

infoServerAccess

```
protected static IRemoteRepository infoServerAccess = null
```

Remote InfoServer Access Repository for loading / saving User Agents and Sources. Accessed via RMI

jdbcConn

```
protected static Connection jdbcConn
```

JDBC Connection for querying provisioning information from database

MainApp

```
public static String MainApp = QNetPQAResource.getString("MainAppPath")
```

Constructor Detail

PQAServletMain

```
public PQAServletMain()
```

Operation Detail

cursorAvailable

```
protected static boolean cursorAvailable()
```

Check to see if cursor limit has been reached

getDbConnection

```
public static Connection getDbConnection()
```

Return a reference to jdbcConn if the CursorLimit has not been exceeded and a valid connection can be established, otherwise return null. This protects from exceeding the a given Database Implementation's cursor limit setting. It is the calling procedures responsibility to call release cursor when finished using the jdbcConn reference. Throws exception if unable to establish a valid database connection

Throws:

SQLException

getRMIConnection

```
public static IRemoteRepository getRMIConnection()
```

Connect the remote object to the server if necessary and return RMI Connection

Throws:

Exception

getWebPagePrinter

```
public WebPagePrinter getWebPagePrinter(WebPage wp)
```

Connect the remote object to the server if necessary and return RMI Connection

Throws:

Exception

loadSection

```
protected void loadSection(HttpServletRequest request, HttpServletResponse response)
```

Evaluate the 'Section' value and load the appropriate page. If Section is not set, then just load the MONTHVIEW_PAGE page

makeDbConnection

```
protected static void makeDbConnection()
```

Make sure jdbcConn is valid. If not, report the error to out and return false

Throws:
SQLException

makeRMIConnection

```
protected static void makeRMIConnection()
```

Connect the remote object to the server if necessary

Throws:
Exception

releaseCursor

```
public static void releaseCursor()
```

Decrement cursorCount so that another process may reference jdbcConn

showDBError

```
protected static void showDBError(HttpServletResponse response)
```

Display a spiffy RMI error message

showRMIError

```
protected static void showRMIError(HttpServletResponse response)
```

Display a spiffy RMI error message

validateDbConnection

```
protected static void validateDbConnection()
```

Make sure con is valid. If not, report the error to out and throw the exception

Throws:
SQLException

validateRMIConnection

```
protected static void validateRMIConnection()
```

make connection if necessary; test the connection; if failed, try to to remake the connection return true if good connection

Class *tcs.ain.infoserver.pqa.QNetPQAResource*

```
public class QNetPQAResource
```

This class is a singleton for a property resource bundle to be used by the QueryNet PQA Servlet. The resource bundle contains values that could potentially be displayed to users. It also provides a direct, static method for safely getting String values without worrying about MIssingResourceExceptions ras; 2/29/99; modified by S. Levine on 5/01/00

Attributes

resourceBundle

Operations

getBundle()
getInt(String)
getString(String)
initResourceBundle()

Attribute Detail

resourceBundle

```
private static ResourceBundle resourceBundle = null
```

The resource bundle we keep for use by the package.

Operation Detail

getBundle

```
public static ResourceBundle getBundle()
```

Return the single, static instance of the resource bundle for the SMPP Module

getInt

```
public static int getInt(String key)
```

Pass through to the resourceBundle, except that we internally catch the MissingResourceExceptıon and report the error but return a "?". Return Int value.

getString

```
public static String getString(String key)
```

Pass through to the resourceBundle, except that we internally catch the MissingResourceException and report the error but return a "?"

initResourceBundle

```
private static void initResourceBundle()
```

Create the resource bundle

Class *tcs.ain.infoserver.QNConst*

```
public class QNConst
```

Attributes

ACTION
ADDBTNTAG
ADDCUSTBTNTAG
ADDCUSTDS
ADDEDITPATH
ADDPUBBTNTAG
ADDPUBDS
AGENTDESC
AGENTDESCTAG
AGENTNAME
AGENTNAMETAG
ALT_ADDRESSING
AVAILABLELIST
AVAILDESTLIST
BTN_ADDCUSTOM
BTN_ADDDEST
BTN_ADDPUBLIC
BTN_CANCEL
BTN_EDITAGENT
BTN_FINISH
BTN_HELP
BTN_LOGIN
BTN_NEXT
BTN_PREV
BTN_REMOVE
BTN_REMOVEDEST
BTN_SELECT
BTN_SUBMIT
CANCELBTNTAG
CBX_PREFIX

CFGCONTENTTAG
CFGCUSTPATH
CREATENEWDS
CURRENTLIST
CUSTAGENTPATH
CUSTDSPATH
CUSTOMIZE
DATASOURCES
DATAWORKER
DEBUGMODE
DEFAULT_FORMAT
DESTINATION
DESTINATIONS
DESTOPTSTAG
DESTPATH
DOM
DOMOPTSTAG
DOW
DOWOPTSTAG
DSNAMETAG
DSOPTSTAG
EDITAGENTBTNTAG
EDT_PREFIX
EDT_WORKERNAME
ERRORMSGTAG
FINISHBTNTAG
FORMNAME
HELPBTNTAG
HOUR
HOUROPTSTAG
HTTP
LOGINBTNTAG
LOGINPATH
MAILBOX
MENUPATH
MINOPTSTAG
MINUTES
MIN_LENGTH
NEXTBTNTAG
NOTIFIER
NOTIFYTAG
ON
ONCHANGEBOX
PIN_LENGTH
PIN_PREFIX
PREVBTNTAG
PREVIEWBTNTAG
PUBDSPATH
REMOVEBTNTAG
REMOVEDESTBTNTAG
REQUIRED
SCRIPTTAG
SELECTBTNTAG
SELECTDESTSTAG
SERVICESTAG
SOURCENAME
START
STATUSPATH
STEP
STYLE
STYLETAG
SUBMITBTNTAG
TESTID
TESTMODE
USERAGENT
USERNAMETAG
VIEWPATH
WORKERDESCTAG
WORKERNAMETAG
WORKERSLIST
WORKOPTSTAG

Attribute Detail

ACTION
```
public static final String ACTION = "action"
```

ADDBTNTAG
```
public static final String ADDBTNTAG = "<!-- ADDBTN -->"
```

ADDCUSTBTNTAG
```
public static final String ADDCUSTBTNTAG = "<!-- ADDCUSTBTN -->"
```

ADDCUSTDS
```
public static final String ADDCUSTDS = "CDS"
```

ADDEDITPATH
```
public static final String ADDEDITPATH = QNServletResource.getString("AddEditPath")
```

ADDPUBBTNTAG
```
public static final String ADDPUBBTNTAG = "<!-- ADDPUBBTN -->"
```

ADDPUBDS
```
public static final String ADDPUBDS = "PDS"
```

AGENTDESC
```
public static final String AGENTDESC = "agentdesc"
```

AGENTDESCTAG
```
public static final String AGENTDESCTAG = "<!-- AGENTDESCINFO -->"
```

AGENTNAME
```
public static final String AGENTNAME = "agentname"
```

AGENTNAMETAG
```
public static final String AGENTNAMETAG = "<!-- AGENTNAMEINFO -->"
```

ALT_ADDRESSING
```
public static final String ALT_ADDRESSING = QNServletResource.getString("supportAltAddressing")
```

AVAILABLELIST
```
public static final String AVAILABLELIST = "availableDSList"
```

AVAILDESTLIST
```
public static final String AVAILDESTLIST = "availDestList"
```

BTN_ADDCUSTOM
```
public static final String BTN_ADDCUSTOM = QNServletResource.getString("AddCustom")
```

BTN_ADDDEST
```
public static final String BTN_ADDDEST = QNServletResource.getString("AddBtn")+ " ->"
```

BTN_ADDPUBLIC
```
public static final String BTN_ADDPUBLIC = QNServletResource.getString("AddPublic")
```

BTN_CANCEL

```
public static final String BTN_CANCEL = QNServletResource.getString("CancelBtn")
```

BTN_EDITAGENT

```
public static final String BTN_EDITAGENT = QNServletResource.getString("EditAgentBtn")
```

BTN_FINISH

```
public static final String BTN_FINISH = QNServletResource.getString("FinishBtn")
```

BTN_HELP

```
public static final String BTN_HELP = QNServletResource.getString("HelpBtn")
```

BTN_LOGIN

```
public static final String BTN_LOGIN = QNServletResource.getString("LoginBtn")
```

BTN_NEXT

```
public static final String BTN_NEXT = QNServletResource.getString("NextBtn")
```

BTN_PREV

```
public static final String BTN_PREV = QNServletResource.getString("PrevBtn")
```

BTN_REMOVE

```
public static final String BTN_REMOVE = QNServletResource.getString("RemoveBtn")
```

BTN_REMOVEDEST

```
public static final String BTN_REMOVEDEST = "<- " + QNServletResource.getString("RemoveBtn")
```

BTN_SELECT

```
public static final String BTN_SELECT = QNServletResource.getString("SelectBtn")
```

BTN_SUBMIT

```
public static final String BTN_SUBMIT = QNServletResource.getString("SubmitBtn")
```

CANCELBTNTAG

```
public static final String CANCELBTNTAG = "<!-- CANCELBTN -->"
```

CBX_PREFIX

```
public static final String CBX_PREFIX = "CBX_"
```

CFGCONTENTTAG

```
public static final String CFGCONTENTTAG = "<!-- CFGCONTENTINFO -->"
```

CFGCUSTPATH

```
public static final String CFGCUSTPATH = QNServletResource.getString("CfgCustPath")
```

CREATENEWDS

```
public static final String CREATENEWDS = "NDS"
```

CURRENTLIST

```
public static final String CURRENTLIST = "currentDestList"
```

CUSTAGENTPATH

```
public static final String CUSTAGENTPATH = QNServletResource.getString("CustAgentPath")
```

CUSTDSPATH

```
public static final String CUSTDSPATH = QNServletResource.getString("CustDSPath")
```

CUSTOMIZE

```
public static final String CUSTOMIZE = "CA"
```

DATASOURCES

```
public static final String DATASOURCES = "dataSources"
```

DATAWORKER

```
public static final String DATAWORKER = "dataWorker"
```

DEBUGMODE

```
public static final boolean DEBUGMODE =
Boolean.valueOf(QNServletResource.getString("DebugMode")).booleanValue()
```

DEFAULT_FORMAT

```
public static final String DEFAULT_FORMAT = "default"
```

DESTINATION

```
public static final String DESTINATION = "DF"
```

DESTINATIONS

```
public static final String DESTINATIONS = "destinations"
```

DESTOPTSTAG

```
public static final String DESTOPTSTAG = "<!-- DESTOPTIONS -->"
```

DESTPATH

```
public static final String DESTPATH = QNServletResource.getString("DestPath")
```

DOM

```
public static final String DOM = "dom"
```

DOMOPTSTAG

```
public static final String DOMOPTSTAG = "<!-- DOMOPTIONS -->"
```

DOW

```
public static final String DOW = "dow"
```

DOWOPTSTAG

```
public static final String DOWOPTSTAG = "<!-- DOWOPTIONS -->"
```

DSNAMETAG

```
public static final String DSNAMETAG = "<!-- DSNAMEINFO -->"
```

DSOPTSTAG

```
public static final String DSOPTSTAG = "<!-- DSOPTIONS -->"
```

EDITAGENTBTNTAG

```
public static final String EDITAGENTBTNTAG = "<!-- EDITAGENTBTN -->"
```

EDT_PREFIX
```
public static final String EDT_PREFIX = "EDT_"
```

EDT_WORKERNAME
```
public static final String EDT_WORKERNAME = "workerNameEdt"
```

ERRORMSGTAG
```
public static final String ERRORMSGTAG = "<!-- ERRORMSG -->"
```

FINISHBTNTAG
```
public static final String FINISHBTNTAG = "<!-- FINISHBTN -->"
```

FORMNAME
```
public static final String FORMNAME = "step"
```

HELPBTNTAG
```
public static final String HELPBTNTAG = "<!-- HELPBTN -->"
```

HOUR
```
public static final String HOUR = "hour"
```

HOUROPTSTAG
```
public static final String HOUROPTSTAG = "<!-- HOUROPTIONS -->"
```

HTTP
```
public static final String HTTP = "http://"
```

LOGINBTNTAG
```
public static final String LOGINBTNTAG = "<!-- LOGINBTN -->"
```

LOGINPATH
```
public static final String LOGINPATH = QNServletResource.getString("LoginPath")
```

MAILBOX
```
public static final String MAILBOX = "INBOX"
```

MENUPATH
```
public static final String MENUPATH = QNServletResource.getString("MenuPath")
```

MINOPTSTAG
```
public static final String MINOPTSTAG = "<!-- MINUTEOPTIONS -->"
```

MINUTES
```
public static final String MINUTES = "minutes"
```

MIN_LENGTH
```
public static final String MIN_LENGTH = QNServletResource.getString("minLength")
```

NEXTBTNTAG
```
public static final String NEXTBTNTAG = "<!-- NEXTBTN -->"
```

NOTIFIER

```
public static final String NOTIFIER = "Notifier"
```

NOTIFYTAG

```
public static final String NOTIFYTAG = "<!-- NOTIFYINFO -->"
```

ON

```
public static final String ON = "on"
```

ONCHANGEBOX

```
public static final String ONCHANGEBOX = "OnChangeBox"
```

PIN_LENGTH

```
public static final String PIN_LENGTH = QNServletResource.getString("PinLength")
```

PIN_PREFIX

```
public static final String PIN_PREFIX = QNServletResource.getString("PinPrefix")
```

PREVBTNTAG

```
public static final String PREVBTNTAG = "<!-- PREVBTN -->"
```

PREVIEWBTNTAG

```
public static final String PREVIEWBTNTAG = "<!-- PREVIEWBTN -->"
```

PUBDSPATH

```
public static final String PUBDSPATH = QNServletResource.getString("PubDSPath")
```

REMOVEBTNTAG

```
public static final String REMOVEBTNTAG = "<!-- REMOVEBTN -->"
```

REMOVEDESTBTNTAG

```
public static final String REMOVEDESTBTNTAG = "<!-- REMOVEDESTBTN -->"
```

REQUIRED

```
public static final String REQUIRED = "Req"
```

SCRIPTTAG

```
public static final String SCRIPTTAG = "<!-- SCRIPT -->"
```

SELECTBTNTAG

```
public static final String SELECTBTNTAG = "<!-- SELECTBTN -->"
```

SELECTDESTSTAG

```
public static final String SELECTDESTSTAG = "<!-- SELECTEDDESTS -->"
```

SERVICESTAG

```
public static final String SERVICESTAG = "<!-- SERVICESINFO -->"
```

SOURCENAME

```
public static final String SOURCENAME = "sourceName"
```

START

```
public static final String START = "START"
```

STATUSPATH

```
public static final String STATUSPATH = QNServletResource.getString("StatusPath")
```

STEP

```
public static final String STEP = "step"
```

STYLE

```
public static final String STYLE = "Style"
```

STYLETAG

```
public static final String STYLETAG = "<!-- STYLEINFO -->"
```

SUBMITBTNTAG

```
public static final String SUBMITBTNTAG = "<!-- SUBMITBTN -->"
```

TESTID

```
public static final String TESTID = "username"
```

TESTMODE

```
public static final boolean TESTMODE =
Boolean.valueOf(QNServletResource.getString("TestMode")).booleanValue()
```

USERAGENT

```
public static final String USERAGENT = "userAgent"
```

USERNAMETAG

```
public static final String USERNAMETAG = "<!-- USERNAMEINFO -->"
```

VIEWPATH

```
public static final String VIEWPATH = QNServletResource.getString("ViewPath")
```

WORKERDESCTAG

```
public static final String WORKERDESCTAG = "<!-- WORKERDESCINFO -->"
```

WORKERNAMETAG

```
public static final String WORKERNAMETAG = "<!-- WORKERNAMEINFO -->"
```

WORKERSLIST

```
public static final String WORKERSLIST = "workersList"
```

WORKOPTSTAG

```
public static final String WORKOPTSTAG = "<!-- WORKEROPTIONS -->"
```

Class *tcs.ain.infoserver.QNServerResource*

```
public class QNServerResource
```

This class is a singleton for a property resource bundle to be used by the Query Net Wizard. The resource bundle contains values that could potentially displayed to users. It also provides a direct, static method for safely getting String values without worrying about MIssingResourceExceptions ras; 2/29/99; modified by S. Levine on 9/9/99

Attributes

resourceBundle

Operations

getBundle()
getInt(String)
getString(String)
initResourceBundle()

Attribute Detail

resourceBundle

```
private static ResourceBundle resourceBundle = null
```

The resource bundle we keep for use by the package.

Operation Detail

getBundle

```
public static ResourceBundle getBundle()
```

Return the single, static instance of the resource bundle for the SMPP Module

getInt

```
public static int getInt(String key)
```

Pass through to the resourceBundle, except that we internally catch the MissingResourceException and report the error but return a "?".
Return Int value.

getString

```
public static String getString(String key)
```

Pass through to the resourceBundle, except that we internally catch the MissingResourceException and report the error but return a "?"
Return String value.

initResourceBundle

```
private static void initResourceBundle()
```

Create the resource bundle

Class *tcs.ain.infoserver.QNWebPagePrinter*

```
public class QNWebPagePrinter
```

Extends:

tcs.web.WebPagePrinter

Attributes

printBanner

Constructors

QNWebPagePrinter(tcs.web.WebPage)

Operations

print()
printPageDescription()

Attribute Detail

printBanner

```
protected boolean printBanner
```

Constructor Detail

QNWebPagePrinter

```
public QNWebPagePrinter(tcs.web.WebPage)
```

Operation Detail

print

```
public void print()
```

printPageDescription

```
protected void printPageDescription()
```

Class *tcs.ain.infoserver.QueryEngine*

```
public abstract class QueryEngine
```

Extends:

tcs.ain.infoserver.InfoServerElement

Implements:

tcs.ain.infoserver.IDataSource

Queries external sources for information which can be made available to DataDestination object. As required, this provides its data as an XML stream.

Attributes

address
DefaultMaxQuerySize
DefaultQueryRefreshTime
formatter
lastCheckDate
maxQuerySize
minAllowedQueryInterval
xslDocs

Operations

addXSLDocument(String, org.w3c.dom.Document)
getAddress()
getDefaultMaxQuerySize()
getFormatter()
getLastCheckDate()
getMinAllowedQueryInterval()
getXSLDocCount()
getXslDocs()
getXSLDocument(String)
getXSLName(int)
removeXSLDocument(String)
setAddress(String)
setDefaultMaxQuerySize(long)
setFormatter(DataFormatter)
setLastCheckDate(long)
setMinAllowedQueryInterval(long)
setXslDocs(Hashtable)

Attribute Detail

address

```
protected String address
```

The address of the data to be queried. Format is dependent on type.

DefaultMaxQuerySize

```
protected static long DefaultMaxQuerySize = 102400
```

Default sizein bytes for how much data is to be queried.

DefaultQueryRefreshTime

```
protected static long DefaultQueryRefreshTime = 1000 * 60 * 30
```

The default interval to wait before refreshing any data.

formatter

```
protected DataFormatter formatter = new DataFormatter()
```

The default dataformatter to be used. This will be used in a launch on demand (i.e. "MO") situation where want to retrieve data directly from the Datasource.

lastCheckDate

```
protected Date lastCheckDate = new Date(0)
```

The last time the external source data was queried; how recent this information is.

maxQuerySize

```
protected long maxQuerySize
```

How much data we will retrieve (in bytes) Default is Max_Query_Size

minAllowedQueryInterval

```
protected long minAllowedQueryInterval = DefaultQueryRefreshTime
```

xslDocs

```
protected Hashtable xslDocs = new Hashtable()
```

Hashtable of XSL Documents, keyed on the descriptive name

Operation Detail

addXSLDocument

```
public void addXSLDocument(String name, org.w3c.dom.Document doc)
```

Add this name/document to the list of offered XSL docs

getAddress

```
public String getAddress()
```

tgGet

getDefaultMaxQuerySize

```
public long getDefaultMaxQuerySize()
```

tgGet

getFormatter

```
public DataFormatter getFormatter()
```

tgGet

getLastCheckDate

```
public Date getLastCheckDate()
```

tgGet

getMinAllowedQueryInterval

```
public long getMinAllowedQueryInterval()
```

tgGet

getXSLDocCount

```
public int getXSLDocCount()
```

return the number of XSL docs that have been assigned

getXslDocs

```
public Hashtable getXslDocs()
```

Format the xslDocs Hashtable for use with the XMLPersister. to do this, the Document object must be converted to its xml string representation

getXSLDocument

```
public org.w3c.dom.Document getXSLDocument(String name)
```

return the XSLDocument assoc with this name. Null if none exists.

getXSLName

```
public String getXSLName(int idx)
```

Return the Name associated with the given index position (0 based) in the hashtable. Null if not available

removeXSLDocument

```
public void removeXSLDocument(String name)
```

Remove the XSL document with this name from the list

setAddress

```
public void setAddress(String address)
```

tgSet

setDefaultMaxQuerySize

```
public void setDefaultMaxQuerySize(long maxQuerySize)
```

tgSet

setFormatter

```
public void setFormatter(DataFormatter formatter)
```

tgSet

setLastCheckDate

```
public void setLastCheckDate(long time)
```

tgset

setMinAllowedQueryInterval

```
public void setMinAllowedQueryInterval(long minAllowedQueryInterval)
```

tgSet

setXslDocs

```
public void setXslDocs(Hashtable ht)
```

Set the xslDocs Hashtable. It must be reformated because the getXslDocs() method reformats it for use with the XMLpersister. We must convert all hashtable string values back into Document objects

Class *tcs.ain.infoserver.QueryNetResource*

```
public class QueryNetResource
```

This class is a singleton for a property resource bundle to be used by the Query Net Wizard. The resource bundle contains values that could potentially displayed to users. It also provides a direct, static method for safely getting String values without worrying about MIssingResourceExceptions ras; 2/29/99; modified by S. Levine on 9/9/99

Attributes

resourceBundle

Operations

getBundle()
getString(String)
initResourceBundle()

Attribute Detail

resourceBundle

```
private static ResourceBundle resourceBundle = null
```

The resource bundle we keep for use by the package.

Operation Detail

getBundle

```
public static ResourceBundle getBundle()
```

Return the single, static instance of the resource bundle for the SMPP Module

getString

```
public static String getString(String key)
```

Pass through to the resourceBundle, except that we internally catch the MissingResourceException and report the error but return a "?"

initResourceBundle

```
private static void initResourceBundle()
```

Create the resource bundle

Class *tcs.ain.infoserver.Repository*

```
public class Repository
```

Class for saving and retrieving UserAgents and QueryEngines. All of the elements owned by a UserAgent will be saved along with the object. This repository just used serialization for storage. More robust storage will be required in future versions.

Attributes

CrsrLogIntrv
cursorCount
cursorLimit
jdbcConn
QueryEngineDbName
queryEngineHash
userAgentHash

Constructors

Repository()

Operations

addQueryEngine(QueryEngine)
addUserAgent(UserAgent, boolean)
cursorAvailable()
getDbConnection()

getQueryEngine(String)
getQueryEngine(int)
getQueryEngineCount()
getQueryEngines()
getUserAgent(String)
getWorkerIdsFromAgent(String)
launchWorker(String, String)
launchWorker(String, String, Hashtable)
loadQueryEngines()
loadUserAgents()
makeDbConnection()
releaseCursor()
saveQueryEngines()
saveUserAgent(UserAgent)
startWorkers()
validateDbConnection()

Attribute Detail

CrsrLogIntrv

```
protected static final int CrsrLogIntrv = QNServerResource.getInt("DbCursorLogIntrv")
```

Amount of tries for a cursor to do before sending "WAITING FOR CURSOR" log message to stdout.

cursorCount

```
protected int cursorCount = 0
```

The database cursors currently open by repository.

cursorLimit

```
protected static final int cursorLimit = QNServerResource.getInt("DbCursorLimit")
```

The database cursor limit for the Repository database connection.

jdbcConn

```
protected Connection jdbcConn
```

The Repositories one and only database connection.

QueryEngineDbName

```
public static final String QueryEngineDbName = "QueryEngine.so"
```

Default name for the QueryEngine peristent storage

queryEngineHash

```
protected Hashtable queryEngineHash = new Hashtable()
```

QueryEngine(DataSource) table. Key is on QueryEngine Id

userAgentHash

```
protected Hashtable userAgentHash = new Hashtable()
```

Agents table. Key is on Agent Id

Constructor Detail

Repository

```
public Repository()
```

Constructor

Operation Detail

addQueryEngine

```
public void addQueryEngine(QueryEngine ds)
```

Add a new QueryEngine to repository

Throws:
InfoServerException

Parameter doc:
ds QueryEngine(DataSource) to add

Exception doc:
tcs.ain.infoserver.InfoServerException

addUserAgent

```
public void addUserAgent(UserAgent ua, boolean bStart)
```

This adds a user agent to the repository. It does not necessary save the agent permanently. Call saveUserAgents for that. ua: UserAgent to add to system. bStart: whether or not to start new agent

Throws:
InfoServerException

Parameter doc:
ua User Agent Object
bStart

Exception doc:
tcs.ain.infoserver.InfoServerException

cursorAvailable

```
private boolean cursorAvailable()
```

Check to see if cursor limit has been reached

getDbConnection

```
public Connection getDbConnection()
```

Return a reference to jdbcConn if the CursorLimit has not been exceeded and a valid connection can be established, otherwise return null. This protects from exceeding the a given Database Implementation's cursor limit setting. It is the calling procedures responsibility to call release cursor when finished using the jdbcConn reference. Throws exception if unable to establish a valid database connection

Throws:
SQLException

getQueryEngine

```
public QueryEngine getQueryEngine(String id)
```

return the data source with this name, null

Parameter doc:
id QueryEngine Descriptor

Return doc:
QueryEngine(DataSource) Instance.

getQueryEngine

```
public QueryEngine getQueryEngine(int idx)
```

Get the data source with this index (0 based).

Throws:
InfoServerException

Parameter doc:
idx DataSource Index

Return doc:
QueryEngine(DataSource) instance.

Exception doc:
tcs.ain.infoserver.InfoServerException

getQueryEngineCount

```
public int getQueryEngineCount()
```

Return the number of QueryEngines available from this repository.

Return doc:
QueryEngine Instance

getQueryEngines

```
public Hashtable getQueryEngines()
```

ReturnsanenumerationofallavailableQueryEngines.

Return doc:
List of all QueryEngines.

getUserAgent

```
public UserAgent getUserAgent(String id)
```

Return the useragent with this ID. Return null if not found.

Parameter doc:
id Agent Descriptor

Return doc:
User Agent Instance.

getWorkerIdsFromAgent

```
public Vector getWorkerIdsFromAgent(String agentId)
```

Return all id's associated with a specified agent

Parameter doc:
agentId Agent Descriptor

Return doc:
List of worker descriptors.

launchWorker

```
public String launchWorker(String agentId, String workerId)
```

Call the specidied workers doWork method this should be used in conjunction with the getWorkerIdsFromAgent() method. this method should only be called if agent and corresponding worker are known to exists!!!

Parameter doc:
agentId Agent Descriptor
workerId Worker Descriptor

Return doc:
Query Results.

launchWorker

```
public String launchWorker(String agentId, String workerId, Hashtable htParams)
```

Call the specidied workers doWork method this should be used in conjunction with the getWorkerIdsFromAgent() method. this method should only be called if agent and corresponding worker are known to exists!!!

Parameter doc:
agentId Agent Descriptor
workerId Worker Descriptor

Return doc:
Query Results.

loadQueryEngines

```
public void loadQueryEngines()
```

Throws:
InfoServerException

loadUserAgents

```
public void loadUserAgents()
```

load UserAgent objects from USERAGENTS table and store in HashTable

Throws:
InfoServerException

makeDbConnection

```
private void makeDbConnection()
```

Make sure jdbcConn is valid. If not, report the error to out and return false

releaseCursor

```
public void releaseCursor()
```

Decrement cursorCount so that another process may reference jdbcConn

saveQueryEngines

```
public void saveQueryEngines()
```

Save New QueryEngine(DataSource) to DB.

Throws:
InfoServerException

Exception doc:
tcs.ain.infoserver.InfoServerException

saveUserAgent

```
public void saveUserAgent(UserAgent ua)
```

Serialize each user agent and store in binary field of USERAGENTS table)

Throws:
InfoServerException

Parameter doc:
ua User Agent Object to Save

Exception doc:
tcs.ain.infoserver.InfoServerException

startWorkers

```
protected void startWorkers()
```

Start each worker for each agent that was running when saved; called by loadUserAgents to get things back to where they were when saved

validateDbConnection

```
private void validateDbConnection()
```

Make sure con is valid. If not, report the error to out and throw the exception

Throws:
SQLException

Class *tcs.ain.infoserver.SearchFilter*

```
public class SearchFilter
```

Extends:
tcs.ain.infoserver.DataFilter

Implements:
tcs.parser.SimpleSaxHandler

Attributes

attrStr
ba
bContinue
endTags
isValidContent
newContent
tag

Operations

assignValue(String)
endElement(String)
getAttrStr()
getSaveables()
getTag()
getXMLStream()
moFilter(DataEvent, Hashtable)
notifyDestinations(DataWorker, IDataSource)
processData(DataEvent)
setAttrStr(String)
setSaveables(Object[])
setTag(String)
startElement(String, String)

Attribute Detail

attrStr

```
protected String attrStr
```

Attribute String to look for

ba

```
protected ByteArrayInputStream ba
```

Filtered Data Stream

bContinue

```
protected boolean bContinue = true
```

Continue comparisons

endTags

```
protected Vector endTags = new Vector()
```

Used to determine when we have left the relm of desired content

isValidContent

```
protected boolean isValidContent = false
```

Found desired content

newContent

```
protected StringBuffer newContent = new StringBuffer()
```

Store filtered content

tag

```
protected String tag
```

Tag to search for

Operation Detail

assignValue

```
public void assignValue(String value)
```

Called when a value within a tag set is processed.

endElement

```
public void endElement(String name)
```

Called when an tag element closes out

getAttrStr

```
public String getAttrStr()
```

getSaveables

```
public Object[] getSaveables()
```

Return an array of all objects that should be persistent. primitive datatypes must be converted to their Object equivalent. These objects will come back in the same order when setSaveables is called.

getTag

```
public String getTag()
```

getXMLStream

```
public InputStream getXMLStream()
```

Return an input stream by which an XML document can be read. We return an INputStream rather than just a DOM XML document because: - some data may be too large to process as a single unit - this way multiple XML parsing techniques (SAX, DOM, etc) can be used at the discretion of the client. The XML Stream does NOT included DTD info.

Throws:
InfoServerException

moFilter

```
public String moFilter(DataEvent evt, Hashtable htParams)
```

Filter data from event and return

notifyDestinations

```
protected void notifyDestinations(DataWorker dw, IDataSource ds)
```

Execute processData() method of all member Destinations

Throws:
InfoServerException

processData

```
public void processData(DataEvent evt)
```

I have new data; Check to see if checksum value for data has changed

setAttrStr

```
public void setAttrStr(String attrStr)
```

setSaveables

```
public void setSaveables(Object[] oa)
```

receive an array of objects that have been saved. the order will correspond to that specified when getSaveables was called. Users should handle Null values in the array.

setTag

```
public void setTag(String tag)
```

startElement

```
public void startElement(String name, String paramStr)
```

Called when a new tag element is being provided. elementName is the (capitalized) tag, and paramStr is the optional parameter list

Class *tcs.ain.infoserver.SearchFilter.XMLInputStream*

```
class XMLInputStream
```

Extends:

java.io.InputStream

This handles access to the XML data through an input stream. Each object to request the inputstream gets one of these, which handles sending the data to the client from the bytearray data. Can be used by children at their discretion (e.g., if spooling cached data)

Attributes

lastPos

Operations

read()

Attribute Detail

lastPos

```
int lastPos = -1
```

Operation Detail

read

```
public int read()
```

Throws:

IOException

Class *tcs.ain.infoserver.servlet.AddEditServicesPage*

```
public class AddEditServicesPage
```

Extends:

tcs.ain.infoserver.servlet.InfoSvrPage

Attributes

access
action
agent
step
stepList
testId

Constructors

AddEditServicesPage(HttpServletRequest, HttpServletResponse, IRemoteRepository, UserAgent)

Operations

cleanUpSession()
doCustAgentStep()
doCustSourceStep()
doDestStep()
doIntroStep()
doNewSourceStep()
doPublicSourceStep()
doSelectWorkerStep()
getBody()
GetPageId()
processWizardForm(DataWorkerWizardForm, boolean, boolean)
syncStepList(Vector, String)

Attribute Detail

access

```
protected IRemoteRepository access
```

action

```
protected String action
```

agent

```
protected UserAgent agent
```

step

```
protected String step = null
```

stepList

```
protected Vector stepList
```

testId

```
protected String testId
```

Constructor Detail

AddEditServicesPage

```
public AddEditServicesPage(HttpServletRequest request, HttpServletResponse response,
IRemoteRepository access, UserAgent agent)
```

Constructor calls super() and sets the title for this page

Operation Detail

cleanUpSession

```
protected void cleanUpSession()
```

doCustAgentStep

```
protected void doCustAgentStep()
```

doCustSourceStep

```
protected void doCustSourceStep()
```

doDestStep

```
protected void doDestStep()
```

doIntroStep

```
protected void doIntroStep()
```

doNewSourceStep

```
protected void doNewSourceStep()
```

doPublicSourceStep

```
protected void doPublicSourceStep()
```

doSelectWorkerStep

```
protected void doSelectWorkerStep()
```

getBody

```
public String getBody()
```

Return the page's body. For increased efficiency, this directly writes to the response output and does not actually return any data.

GetPageId

```
public static int GetPageId()
```

processWizardForm

```
protected String processWizardForm(DataWorkerWizardForm wizardForm, boolean doProcess, boolean
doPrint)
```

syncStepList

```
protected void syncStepList(Vector list, String currStep)
```

Return the page's body. For increased efficiency, this directly writes to the response output and does not actually return any data.

Class *tcs.ain.infoserver.servlet.CreateNewSourceForm*

```
public class CreateNewSourceForm
```

Extends:

tcs.ain.infoserver.servlet.DataWorkerWizardForm

Define the Data Sources that are desired.

Attributes

debug
HtmlPage
request
testId

Constructors

CreateNewSourceForm(HttpServletRequest, IRemoteRepository, String)

Operations

getCustomSource(String, StringBuffer)
getForm()
getSource()
getTagTable()
populateFormFromWorker()
populateWorkerFromForm()
printCustomForm(String)
processCustomSource(StringBuffer)
setDebug(boolean)

Attribute Detail

debug

```
private boolean debug = true
```

For Internal Debugging

HtmlPage

```
protected static StringBuffer HtmlPage
```

Stores Html Page associated with a form producing class

request

```
protected HttpServletRequest request
```

HttpServletRequest

testId

```
protected String testId
```

Stores error messages recieved from creating a new source

Constructor Detail

CreateNewSourceForm

```
public CreateNewSourceForm(HttpServletRequest request, IRemoteRepository access, String stepDesc)
```

Constructor

Operation Detail

getCustomSource

```
protected QueryEngine getCustomSource(String source, StringBuffer errorMsg)
```

Get a new Custom DataSource from the ISElementCreator Implementation

getForm

```
public String getForm()
```

Return the appropriate form for adding new workers or selecting existing workers

getSource

```
protected String getSource()
```

get selected source name from request object

getTagTable

```
protected Hashtable getTagTable()
```

Create a table of tags and their values to be used in replacing tags found in static html pages. Used to add dynamic content to these pages.

populateFormFromWorker

```
protected void populateFormFromWorker()
```

Populate the form fields with values from the DataWorker

populateWorkerFromForm

```
public String populateWorkerFromForm()
```

Populate the DataWorker with data assigned to the form

printCustomForm

```
protected String printCustomForm(String source)
```

processCustomSource

```
protected void processCustomSource(StringBuffer errorMsg)
```

Handle logic flow of creating a new custom datasource

setDebug

```
public void setDebug(boolean b)
```

Return the appropriate form for adding new workers or selecting existing workers

Class *tcs.ain.infoserver.servlet.CustomizeAgentForm*

```
public class CustomizeAgentForm
```

Extends:

tcs.ain.infoserver.servlet.DataWorkerWizardForm

Define the Data Sources that are desired.

Attributes

desc
dom
domIndex
domTable
dow
dowIndex
dowTable
hour
hourIndex
hourTable
HtmlPage
minute
minuteIndex
minuteTable
newId
testId

Constructors

CustomizeAgentForm(HttpServletRequest, IRemoteRepository, String)

Operations

addTimerValues(String[], Hashtable, TimeElement)
createDomIndex()
createDomTable()
createDowIndex()
createDowTable()
createHourIndex()
createHourTable()
createIndex(Vector, String)
createMinuteIndex()
createMinuteTable()
getForm()
getTagTable()
populateFormFromWorker()
populateWorkerFromForm()
saveToRepository()
setAgentPropeties()
setCustomTimer()
writeSelectBox(String, Hashtable, Vector, TimeElement)

Attribute Detail

desc

protected String desc

dom

protected String[] dom

domIndex

protected static Vector domIndex = null

domTable

protected static Hashtable domTable = null

dow

protected String[] dow

dowIndex

protected static Vector dowIndex = null

dowTable

protected static Hashtable dowTable = null

hour

protected String[] hour

hourIndex

protected static Vector hourIndex = null

hourTable

protected static Hashtable hourTable = null

HtmlPage

protected static StringBuffer HtmlPage

Stores Html Page associated with a form producing class

minute

protected String[] minute

minuteIndex

protected static Vector minuteIndex = null

minuteTable

protected static Hashtable minuteTable = null

newId

protected String newId

testId

protected String testId

Constructor Detail

CustomizeAgentForm

public CustomizeAgentForm(HttpServletRequest request, IRemoteRepository access, String stepDesc)

Constructor

Operation Detail

addTimerValues

```
protected void addTimerValues(String[] timerValues, Hashtable table, TimeElement timeElement)
```

createDomIndex

```
protected static void createDomIndex()
```

createDomTable

```
protected static void createDomTable()
```

Create a table to hold Day of Month options to be displayed to user as well as the corresponding TimeElement values

createDowIndex

```
protected static void createDowIndex()
```

createDowTable

```
protected static void createDowTable()
```

Create a table to hold Day of Week options to be displayed to user as well as the corresponding TimeElement values

createHourIndex

```
protected static void createHourIndex()
```

createHourTable

```
protected static void createHourTable()
```

Create a table to hold Hout options to be displayed to user as well as the corresponding TimeElement values

createIndex

```
protected static void createIndex(Vector vIndex, String property)
```

..... /create vector list of possible index value for the specified index from the specified /resource string.

createMinuteIndex

```
protected static void createMinuteIndex()
```

createMinuteTable

```
protected static void createMinuteTable()
```

Create a table to hold Minute options to be displayed to user as well as the corresponding TimeElement values

getForm

```
public String getForm()
```

Return the appropriate form for adding new workers or selecting existing workers

getTagTable

```
protected Hashtable getTagTable()
```

Create a table of tags and their values to be used in replacing tags found in static html pages. Used to add dynamic content to these pages.

populateFormFromWorker

```
protected void populateFormFromWorker()
```

Populate the form fields with values from the DataWorker

populateWorkerFromForm

```
public String populateWorkerFromForm()
```

Populate the DataWorker with data assigned to the form

saveToRepository

```
public void saveToRepository()
```

setAgentPropeties

```
protected void setAgentPropeties()
```

setCustomTimer

```
protected String setCustomTimer()
```

writeSelectBox

```
protected String writeSelectBox(String name, Hashtable table, Vector index, TimeElement
timeElement)
```

Class *tcs.ain.infoserver.servlet.CustomSourceForm*

```
public class CustomSourceForm
```

Extends:

tcs.ain.infoserver.servlet.DataWorkerWizardForm

Define the Data Sources that are desired.

Attributes

HtmlPage
request

Constructors

CustomSourceForm(HttpServletRequest, IRemoteRepository, String)

Operations

getAvailableDS()
getForm()
getTagTable()
populateFormFromWorker()
populateWorkerFromForm()

Attribute Detail

HtmlPage

```
protected static StringBuffer HtmlPage
```

Stores Html Page associated with a form producing class

request

```
protected HttpServletRequest request
```

Constructor Detail

CustomSourceForm

```
public CustomSourceForm(HttpServletRequest request, IRemoteRepository access, String stepDesc)
```

Constructor

Operation Detail

getAvailableDS

```
protected String getAvailableDS()
```

create select box for available datasources and store in buffer

getForm

```
public String getForm()
```

Return the appropriate form for adding new workers or selecting existing workers

getTagTable

```
protected Hashtable getTagTable()
```

Create a table of tags and their values to be used in replacing tags found in static html pages. Used to add dynamic content to these pages.

populateFormFromWorker

```
protected void populateFormFromWorker()
```

Populate the form fields with values from the DataWorker

populateWorkerFromForm

```
public String populateWorkerFromForm()
```

Edit Data Worker based on user request

Class tcs.ain.infoserver.servlet.DataSourceForm

```
public class DataSourceForm
```

Extends:

tcs.ain.infoserver.servlet.DataWorkerWizardForm

Attributes

request

Constructors

DataSourceForm(javax.servlet.http.HttpServletRequest, tcs.ain.infoserver.access.IRemoteRepository, java.lang.String)

Operations

getAvailableDS()
getForm()
parseComment(java.lang.String)
parseDataSource(java.lang.String)
populateFormFromWorker()
populateWorkerFromForm()
processDataSource()

Attribute Detail

request

```
protected javax.servlet.http.HttpServletRequest request
```

Constructor Detail

DataSourceForm

```
public DataSourceForm(javax.servlet.http.HttpServletRequest,
tcs.ain.infoserver.access.IRemoteRepository, java.lang.String)
```

Operation Detail

getAvailableDS

```
protected void getAvailableDS()
```

getForm

```
public java.lang.String getForm()
```

parseComment

```
protected java.lang.String parseComment(java.lang.String)
```

parseDataSource

```
protected java.lang.String parseDataSource(java.lang.String)
```

populateFormFromWorker

```
protected void populateFormFromWorker()
```

populateWorkerFromForm

```
public java.lang.String populateWorkerFromForm()
```

processDataSource

```
public void processDataSource()
```

Class tcs.ain.infoserver.servlet.DataWorkerForm

```
public class DataWorkerForm
```

Extends:

tcs.ain.infoserver.servlet.DataWorkerWizardForm

Define the Data Sources that are desired.

Attributes

HtmlPage
selectWorker
testId

Constructors

DataWorkerForm(HttpServletRequest, IRemoteRepository, String)

Operations

getForm()
getTagTable()
getWorkerOptions()
getWorkersForm()
populateFormFromWorker()
populateWorkerFromForm()

Attribute Detail

HtmlPage

```
protected static StringBuffer HtmlPage
```

Stores Html Page associated with a form producing class

selectWorker

```
protected String selectWorker
```

Name of Selected worker to be edited

testId

```
protected String testId
```

Id to be used for script testing only

Constructor Detail

DataWorkerForm

```
public DataWorkerForm(HttpServletRequest request, IRemoteRepository access, String stepDesc)
```

Constructor

Operation Detail

getForm

```
public String getForm()
```

Return the appropriate form for adding new workers or selecting existing workers

getTagTable

```
protected Hashtable getTagTable()
```

Create a table of tags and their values to be used in replacing tags found in static html pages. Used to add dynamic content to these pages.

getWorkerOptions

```
protected String getWorkerOptions()
```

getWorkersForm

```
protected void getWorkersForm()
```

Create an html form for selecting or removing existing Data Workers. This is the initial form for the Query Net Wizard

populateFormFromWorker

```
protected void populateFormFromWorker()
```

Populate the form fields with values from the DataWorker

populateWorkerFromForm

```
public String populateWorkerFromForm()
```

Populate the DataWorker with data assigned to the form

Class *tcs.ain.infoserver.servlet.DataWorkerWizardForm*

```
public abstract class DataWorkerWizardForm
```

Base class for elements that will show info about a DataWorker Subclasses will access each component that a DataWorker cares about

Attributes

action
agent
buffer
errorMessage
repository
session
testId
worker

Constructors

DataWorkerWizardForm(HttpServletRequest, IRemoteRepository, String)

Operations

getForm()
getHtmlPage(String)
populateFormFromWorker()
populateWorkerFromForm()
printJavaScript()
replaceTags(String, Hashtable)

Attribute Detail

action

```
protected String action
```

agent

```
protected UserAgent agent
```

buffer

```
protected StringBuffer buffer
```

errorMessage

```
protected String errorMessage = ""
```

repository

```
protected IRemoteRepository repository
```

session

```
protected HttpSession session
```

testId

```
protected String testId
```

Used for testing scripts only

worker

```
protected DataWorker worker
```

Constructor Detail

DataWorkerWizardForm

```
public DataWorkerWizardForm(HttpServletRequest request, IRemoteRepository r, String errorMessage)
```

Constructor

Operation Detail

getForm

```
public abstract String getForm()
```

Return the HTML string that defines the form

getHtmlPage

```
public StringBuffer getHtmlPage(String path)
```

Read in specified Html Page and store to static HtmlPage StringBuffer

Throws:
Exception

populateFormFromWorker

```
protected abstract void populateFormFromWorker()
```

Populate the form fields with values from the DataWorker

populateWorkerFromForm

```
public abstract String populateWorkerFromForm()
```

Populate the DataWorker with data assigned to the form

printJavaScript

```
public String printJavaScript()
```

java script for create pop up browser for displaying help info

replaceTags

```
public String replaceTags(String content, Hashtable ht)
```

Read in specified Html Page and store content to StringBuffer

Throws:

Exception

Class *tcs.ain.infoserver.servlet.DestinationForm*

```
public class DestinationForm
```

Extends:

tcs.ain.infoserver.servlet.DataWorkerWizardForm

Define the Data Sources that are desired.

Attributes

bNotifyChangeOnly
dataSource
destStore
HtmlPage
request
testId

Constructors

DestinationForm(HttpServletRequest, IRemoteRepository, String)

Operations

addDestination(String, IDataEventSource)
addToList()
getDestination(String)
getForm()
getTagTable()
initList()
populateFormFromWorker()
populateWorkerFromForm()
printAvailDestList()
printCurrDestList()
printNotifyChangeBox()
printRequiredInfo(String)
printStyles(String)
printStyleSection()
processAlwaysNotify()
processDestinations()
processNotifyChangeOnly()
removeDestinations(IDataEventSource)
removeFromList()

Attribute Detail

bNotifyChangeOnly

```
protected boolean bNotifyChangeOnly = false
```

dataSource

```
protected QueryEngine dataSource = null
```

destStore

```
protected Vector destStore = null
```

HtmlPage

```
protected static StringBuffer HtmlPage
```

Stores Html Page associated with a form producing class

request

```
protected HttpServletRequest request = null
```

testId

```
protected String testId = null
```

Constructor Detail

DestinationForm

```
public DestinationForm(HttpServletRequest request, IRemoteRepository access, String stepDesc)
```

Constructor

Operation Detail

addDestination

```
protected String addDestination(String destination, IDataEventSource iDes)
```

Add a destination to Data Worker

addToList

```
public void addToList()
```

Add new destinations to destStore Vector and store into session obj

getDestination

```
protected UserDeliveryDestination getDestination(String destId)
```

Return requested destination.

getForm

```
public String getForm()
```

Return the appropriate form for adding new workers or selecting existing workers

getTagTable

```
protected Hashtable getTagTable()
```

Create a table of tags and their values to be used in replacing tags found in static html pages. Used to add dynamic content to these pages.

initList

```
protected void initList()
```

Initialize destStore

populateFormFromWorker

```
protected void populateFormFromWorker()
```

Populate the form fields with values from the DataWorker

populateWorkerFromForm

```
public String populateWorkerFromForm()
```

Populate the DataWorker with data assigned to the form, or temporarily store.

printAvailDestList

```
protected String printAvailDestList()
```

Create an html Select box and populate with available destinations

printCurrDestList

```
protected String printCurrDestList()
```

Create an html Select box and populate with currently selected destinations

printNotifyChangeBox

```
protected String printNotifyChangeBox()
```

Create an html CheckBox for "Notify If Change Only" option

printRequiredInfo

```
protected String printRequiredInfo(String destId)
```

Create a text box and prompt for required info for a given destination

printStyles

```
protected String printStyles(String destId)
```

Create and populate a drop down box with available style options for a give data sources

printStyleSection

```
protected String printStyleSection()
```

Create table rows dynamically based on Data Source info

processAlwaysNotify

```
protected String processAlwaysNotify()
```

Add all destinations to Data Worker

processDestinations

```
public String processDestinations()
```

process selected destinations *

processNotifyChangeOnly

```
protected String processNotifyChangeOnly()
```

Add destinations to NotifyChangeOnlyDestination, then add NotifyChangeOnlyDestination to Worker

removeDestinations

```
protected void removeDestinations(IDataEventSource des)
```

Remove destination from Data Worker

removeFromList

```
public void removeFromList()
```

Remove selected destinations from destStore Vector

Class *tcs.ain.infoserver.servlet.InfoSvrMain*

```
public class InfoSvrMain
```

Extends:

tcs.web.WebMain

This is the Main page for the Info Server Web Application. This loads pages according to the received Section variable

Attributes

CrsrLogIntrv
cursorCount
cursorLimit
infoServerAccess
jdbcConn
MainApp
sessions

Constructors

InfoSvrMain()

Operations

cursorAvailable()
getDbConnection()
getRMIConnection()
getSessionObj(Object)
getWebPagePrinter(WebPage)
loadSection(HttpServletRequest, HttpServletResponse)
makeDbConnection()
makeRMIConnection()
putSessionObj(Object, Object)
releaseCursor()
removeSessionObj(Object)
showDBError(HttpServletResponse)
showRMIError(HttpServletResponse)
validateDbConnection()
validateRMIConnection()

Attribute Detail

CrsrLogIntrv

```
protected static final int CrsrLogIntrv = QNServletResource.getInt("DbCursorLogIntrv")
```

Amount of tries for a cursor to do before sending "WAITING FOR CURSOR" log message to stdout.

cursorCount

```
protected static int cursorCount = 0
```

The database cursors currently open by repository.

cursorLimit

```
protected static final int cursorLimit = QNServletResource.getInt("DbCursorLimit")
```

The database cursor limit for the Repository database connection.

infoServerAccess

```
protected static IRemoteRepository infoServerAccess = null
```

Remote InfoServer Access Repository for loading / saving User Agents and Sources. Accessed via RMI

jdbcConn

```
protected static Connection jdbcConn
```

JDBC Connection for querying provisioning information from database

MainApp

```
public static String MainApp = QNServletResource.getString("MainAppPath")
```

sessions

```
private static Hashtable sessions
```

used for script load testing where regular session objects are not available

Constructor Detail

InfoSvrMain

```
public InfoSvrMain()
```

Operation Detail

cursorAvailable

```
protected static boolean cursorAvailable()
```

Check to see if cursor limit has been reached

getDbConnection

```
public static Connection getDbConnection()
```

Return a reference to jdbcConn if the CursorLimit has not been exceeded and a valid connection can be established, otherwise return null. This protects from exceeding the a given Database Implementation's cursor limit setting. It is the calling procedures responsibility to call release cursor when finished using the jdbcConn reference. Throws exception if unable to establish a valid database connection

Throws:

SQLException

getRMIConnection

```
public static IRemoteRepository getRMIConnection()
```

Connect the remote object to the server if necessary and return RMI Connection

Throws:

Exception

getSessionObj

```
public static Object getSessionObj(Object key)
```

Use this instead of session object when script testing

getWebPagePrinter

```
public WebPagePrinter getWebPagePrinter(WebPage wp)
```

Connect the remote object to the server if necessary and return RMI Connection

Throws:

Exception

loadSection

```
protected void loadSection(HttpServletRequest request, HttpServletResponse response)
```

Evaluate the 'Section' value and load the appropriate page. If Section is not set, then just load the MONTHVIEW_PAGE page

makeDbConnection

```
protected static void makeDbConnection()
```

Make sure jdbcConn is valid. If not, report the error to out and return false

Throws:
SQLException

makeRMIConnection

```
protected static void makeRMIConnection()
```

Connect the remote object to the server if necessary

Throws:
Exception

putSessionObj

```
public static void putSessionObj(Object key, Object value)
```

Use this instead of session object when script testing

releaseCursor

```
public static void releaseCursor()
```

Decrement cursorCount so that another process may reference jdbcConn

removeSessionObj

```
public static Object removeSessionObj(Object key)
```

Use this instead of session object when script testing

showDBError

```
protected static void showDBError(HttpServletResponse response)
```

Display a spiffy RMI error message

showRMIError

```
protected static void showRMIError(HttpServletResponse response)
```

Display a spiffy RMI error message

validateDbConnection

```
protected static void validateDbConnection()
```

Make sure con is valid. If not, report the error to out and throw the exception

Throws:
SQLException

validateRMIConnection

```
protected static void validateRMIConnection()
```

make connection if necessary; test the connection; if failed, try to to remake the connection return true if good connection

Class *tcs.ain.infoserver.servlet.InfoSvrPage*

```
public abstract class InfoSvrPage
```

Extends:
tcs.web.WebPage

Attributes

userAgent

Constructors

InfoSvrPage(HttpServletRequest, HttpServletResponse, UserAgent)

Operations

getHtmlPage(String)
replaceTags(String, Hashtable)

Attribute Detail

userAgent

```
protected UserAgent userAgent
```

Constructor Detail

InfoSvrPage

```
public InfoSvrPage(HttpServletRequest req, HttpServletResponse res, UserAgent agent)
```

Stores Html Page associated with a form producing class

Operation Detail

getHtmlPage

```
public StringBuffer getHtmlPage(String path)
```

Read in specified Html Page and store to static HtmlPage StringBuffer

Throws:
Exception

replaceTags

```
public String replaceTags(String content, Hashtable ht)
```

Read in specified Html Page and store content to StringBuffer

Throws:
Exception

Class *tcs.ain.infoserver.servlet.ISElementCreatorException*

```
public class ISElementCreatorException
```

Extends:
tcs.ain.infoserver.servlet.ISServletException

Constructors

ISElementCreatorException(String)

Constructor Detail

ISElementCreatorException

```
public ISElementCreatorException(String s)
```

Class *tcs.ain.infoserver.servlet.ISElementLoader*

```
public class ISElementLoader
```

Attributes

creator

resourceBundle

Operations

getBundle()
getCreator()
initResourceBundle()

Attribute Detail

creator

```
protected static ISElementCreator creator = null
```

resourceBundle

```
protected static ResourceBundle resourceBundle = null
```

Operation Detail

getBundle

```
private static ResourceBundle getBundle()
```

Return the single, static instance of the resource bundle for the SMPP Module

getCreator

```
public static ISElementCreator getCreator()
```

Throws:

ClassNotFoundException
InstantiationException
IllegalAccessException

initResourceBundle

```
private static void initResourceBundle()
```

Create the resource bundle

Class *tcs.ain.infoserver.servlet.ISServletException*

```
public class ISServletException
```

Extends:

java.lang.Exception

Constructors

ISServletException(String)

Constructor Detail

ISServletException

```
public ISServletException(String s)
```

Class *tcs.ain.infoserver.servlet.LoginPage*

```
public class LoginPage
```

Extends:

tcs.ain.infoserver.servlet.InfoSvrPage

USER LOGIN PAGE. This prompts user for username/pw and tries and to find (or create) a UserAgent object for them. The UserAgent object is inserted into the user's Session under the name 'userAgent', for retention between screens This will eventually validate against and reference the SMS Internet Suite UserInfo database. Note: New agents are created for authenticated users, and their WebProv values will be assigned to the agent for name, email, min We will need to keep the agent info up-to-date somehow.

Attributes

access
HtmlPage
password
testId
username

Constructors

LoginPage(HttpServletRequest, HttpServletResponse, IRemoteRepository)

Operations

createNewAgent(ResultSet)
getBody()
getInvalidLogin()
getLoginForm()
GetPageId()
isValidLogin(ResultSet)
print()
processLogin()
queryUserInfoDB()
syncAgent(ResultSet, UserAgent)

Attribute Detail

access

```
protected IRemoteRepository access
```

How we obtain Agents

HtmlPage

```
protected static StringBuffer HtmlPage
```

Stores Html Page associated with a form producing class

password

```
protected String password
```

testId

```
protected String testId
```

username

```
protected String username
```

Constructor Detail

LoginPage

```
public LoginPage(HttpServletRequest request, HttpServletResponse response, IRemoteRepository
access)
```

Constructor calls super() and sets the title for this page

Operation Detail

createNewAgent

```
protected UserAgent createNewAgent(ResultSet r)
```

Create a new UserAgent based on information we can get from the UserProvisioning database for the login name

getBody

```
public String getBody()
```

Return the page's body

getInvalidLogin

```
protected String getInvalidLogin()
```

Return String describing that the login/pw were invalid

getLoginForm

```
protected String getLoginForm()
```

Show the login form. When accepted, this page will get called again, but then it will process the results

GetPageId

```
public static int GetPageId()
```

isValidLogin

```
protected boolean isValidLogin(ResultSet r)
```

Autherticate the user.

print

```
public void print()
```

Ensure that pagePrint is set to not print banner

processLogin

```
protected String processLogin()
```

process the login; if acceptable, stick in the Session and show Welcome Page

queryUserInfoDB

```
protected ResultSet queryUserInfoDB()
```

Query the UserProvisioning database for the login name

syncAgent

```
protected void syncAgent(ResultSet r, UserAgent agent)
```

Make sure agent info is up to date with provisioning information.

Class *tcs.ain.infoserver.servlet.MainMenuPage*

```
public class MainMenuPage
```

Extends:

tcs.ain.infoserver.servlet.InfoSvrPage

MAIN MENU PAGE. This will provide the main options for the app.

Attributes

HtmlPage
SiteVec

Constructors

MainMenuPage(HttpServletRequest, HttpServletResponse, UserAgent)

Operations

getBody()
getPageDescription()
GetPageId()
getSiteVec()

Attribute Detail

HtmlPage

```
protected static StringBuffer HtmlPage
```

Stores Html Page associated with a form producing class

SiteVec

```
protected static Vector SiteVec = new Vector()
```

This contains a central list of the site account names. Referenced elsewhere, but just populated here

Constructor Detail

MainMenuPage

```
public MainMenuPage(HttpServletRequest request, HttpServletResponse response, UserAgent agent)
```

Constructor calls super() and sets the title for this page

Operation Detail

getBody

```
public String getBody()
```

the page body

getPageDescription

```
public String getPageDescription()
```

This returns the special page description.

GetPageId

```
public static int GetPageId()
```

getSiteVec

```
public static Vector getSiteVec()
```

Return the static vector of available sites

Class *tcs.ain.infoserver.servlet.PublicSourceForm*

```
public class PublicSourceForm
```

Extends:

tcs.ain.infoserver.servlet.DataWorkerWizardForm

Define the Data Sources that are desired.

Attributes

HtmlPage
request
testId

Constructors

PublicSourceForm(HttpServletRequest, IRemoteRepository, String)

Operations

getAvailableDS()
getForm()
getTagTable()
parseDataSource(String)
populateFormFromWorker()

populateWorkerFromForm()
processDataSource()

Attribute Detail

HtmlPage

```
protected static StringBuffer HtmlPage
```

Stores Html Page associated with a form producing class

request

```
protected HttpServletRequest request
```

Request object

testId

```
protected String testId
```

Used for testing via scripts only

Constructor Detail

PublicSourceForm

```
public PublicSourceForm(HttpServletRequest request, IRemoteRepository access, String stepDesc)
```

Constructor

Operation Detail

getAvailableDS

```
protected String getAvailableDS()
```

create select box for available datasources and store in buffer

getForm

```
public String getForm()
```

Return the appropriate form for adding new workers or selecting existing workers

getTagTable

```
protected Hashtable getTagTable()
```

Create a table of tags and their values to be used in replacing tags found in static html pages. Used to add dynamic content to these pages.

parseDataSource

```
protected String parseDataSource(String source)
```

Remove comment from destination. Comment is delimited with ":" in front. ex./ parseDestination("EMAIL: This is a public email destination.") returns "EMAIL"

populateFormFromWorker

```
protected void populateFormFromWorker()
```

Populate the form fields with values from the DataWorker

populateWorkerFromForm

```
public String populateWorkerFromForm()
```

Edit Data Worker based on user request

processDataSource

```
public void processDataSource()
```

Edit Data Worker based on user request

Class *tcs.ain.infoserver.servlet.QNServiceCreator*

```
public class QNServiceCreator
```

Implements:

tcs.ain.infoserver.servlet.ISElementCreator

S. Levine on 11/1/99

Attributes

customSourcesDb
debug
destinationsDb
STARTREF
URL

Constructors

QNServiceCreator()

Operations

getAttribute(String, String, int)
getAttribute(String, String)
getAttribute(String, int, String)
getChatDestination(String, Vector, StringBuffer)
getCustEmailSourceForm(String, InfoServerElement, UserAgent)
getCustomEmailSource(String, HttpServletRequest, StringBuffer)
getCustomSource(String, HttpServletRequest, StringBuffer)
getCustomWebSource(String, HttpServletRequest, StringBuffer)
getCustWebSourceForm(String, InfoServerElement, UserAgent)
getDestination(String, Vector, StringBuffer)
getElementNames(int, boolean)
getEmailDestination(String, Vector, StringBuffer)
getFormatStyle(UserDeliveryDestination)
getFormatter(QueryEngine, String)
getMinFromField(String, String)
getReqChatDestInfo(InfoServerElement, UserAgent)
getReqEmailDestInfo(InfoServerElement, UserAgent)
getReqEmailSourceInfo(InfoServerElement, UserAgent)
getReqSMSDestInfo(InfoServerElement, UserAgent)
getRequiredData(String, InfoServerElement, UserAgent, int)
getRequirements(String, int)
getSimpleDestination(String, StringBuffer)
getSMSDestination(String, Vector, StringBuffer)
getStyles(String, int)
getSubForm(String, InfoServerElement, UserAgent, int)
isValidHostName(String)
loadElements(String, String)
rateReliability(String)
returnDbType(int)
validMin(String)

Attribute Detail

customSourcesDb

```
private Hashtable customSourcesDb = null
```

Store custom source elements.

debug

```
private static final boolean debug = true
```

For Internal Debugging

destinationsDb

```
private Hashtable destinationsDb = null
```

Store Destination elements.

STARTREF

```
private static final String STARTREF = QNConst.EDT_PREFIX + "STARTREF"
```

Name of url field Create Web Source Form

URL

```
private static final String URL = QNConst.EDT_PREFIX + "URL"
```

Name of url field Create Web Source Form

Constructor Detail

QNServiceCreator

```
public QNServiceCreator()
```

Return the single, static instance of the resource bundle for the SMPP Module

Operation Detail

getAttribute

```
protected String getAttribute(String name, String attr, int elementType)
```

Return the attribute from an Element Config String given the element name, attribute, and elementType.

getAttribute

```
protected String getAttribute(String configString, String atName)
```

Return the attribute from an Element Config String given the string, and the attribute.

getAttribute

```
private Vector getAttribute(String name, int elementType, String attr)
```

Return Parsed Attribute String, and return elements in a vector

getChatDestination

```
private ChatUserDestination getChatDestination(String destination, Vector reqInfo, StringBuffer
errorMsg)
```

Load elements descriptions from resource file into hashtables

getCustEmailSourceForm

```
protected String getCustEmailSourceForm(String name, InfoServerElement element, UserAgent agent)
```

Get form components necessary in creating the specifed element

getCustomEmailSource

```
private NewEmailQueryEngine getCustomEmailSource(String source, HttpServletRequest request,
StringBuffer errorMsg)
```

Load elements descriptions from resource file into hashtables

getCustomSource

```
public QueryEngine getCustomSource(String source, HttpServletRequest request, StringBuffer
errorMsg)
```

Load elements descriptions from resource file into hashtables

getCustomWebSource

```
private QueryEngine getCustomWebSource(String source, HttpServletRequest request, StringBuffer errorMsg)
```

Load elements descriptions from resource file into hashtables

getCustWebSourceForm

```
protected String getCustWebSourceForm(String name, InfoServerElement element, UserAgent agent)
```

Get form components necessary in creating a web datasource

getDestination

```
public UserDeliveryDestination getDestination(String destination, Vector reqInfo, StringBuffer errorMsg)
```

Load elements descriptions from resource file into hashtables

getElementNames

```
public Vector getElementNames(int elementType, boolean addDesc)
```

return list of elements descriptions from specified hashtable

getEmailDestination

```
private EmailUserDestination getEmailDestination(String destination, Vector reqInfo, StringBuffer errorMsg)
```

Load elements descriptions from resource file into hashtables

getFormatStyle

```
public String getFormatStyle(UserDeliveryDestination dest)
```

Return the name of the XSL format being used by this destination

getFormatter

```
public DataFormatter getFormatter(QueryEngine ds, String style)
```

Load elements descriptions from resource file into hashtables

getMinFromField

```
protected String getMinFromField(String value, String srchFld)
```

try to lookup and return the MIN from the UserInfo table using the recvd search field returns null if not found

getReqChatDestInfo

```
protected Vector getReqChatDestInfo(InfoServerElement element, UserAgent agent)
```

getReqEmailDestInfo

```
protected Vector getReqEmailDestInfo(InfoServerElement element, UserAgent agent)
```

getReqEmailSourceInfo

```
protected Vector getReqEmailSourceInfo(InfoServerElement element, UserAgent agent)
```

getReqSMSDestInfo

```
protected Vector getReqSMSDestInfo(InfoServerElement element, UserAgent agent)
```

getRequiredData

```
public Vector getRequiredData(String destId, InfoServerElement element, UserAgent agent, int elementType)
```

Return existing Required Data associated with an existing element

getRequirements

```
public Vector getRequirements(String name, int elementType)
```

Return Required Information necessary for creating an element of this type

getSimpleDestination

```
private SimpleUserDestination getSimpleDestination(String destination, StringBuffer errorMsg)
```

Create a Simple user destination

getSMSDestination

```
private SMSUserDestination getSMSDestination(String destination, Vector reqInfo, StringBuffer errorMsg)
```

Load elements descriptions from resource file into hashtables

getStyles

```
protected Vector getStyles(String name, int elementType)
```

Return Required Information necessary for creating an element of this type

getSubForm

```
public String getSubForm(String name, InfoServerElement element, UserAgent agent, int elementType)
```

Get form components necessary in creating the specifed element

isValidHostName

```
protected boolean isValidHostName(String host)
```

Check to see if host name is valid. Return True if valid False if invalid.

loadElements

```
public void loadElements(String sourceElements, String destElements)
```

Load elements descriptions into hashtables, this method should only ever be called once by the loader so no need to synchronize.

rateReliability

```
protected String rateReliability(String key)
```

Relate reliability of Xml parsing key based on a scale of 1-10 were 1=most reliable and 10=least reliable. The less tags in the key, the more reliable. 4 tags or less is a 1, and every tag after that adds 1 to the rating. For ex./ Clip.Html[0].Body[0].Table[0] is very reliable and would be a 1, and Clip.Html[0].Body[0].Table[2].TR[1].TD[2] would be less reliable(but still reasonably reliable) and would be a 3

returnDbType

```
protected Hashtable returnDbType(int elementType)
```

Return correct hashtable dependent on elementType

validMin

```
protected boolean validMin(String address)
```

Look at the msgMIN field to ensure that it really contains a MIN. It is possible that it could actually contain alternate addressing info, which must be translated to a MIN address.

Class *tcs.ain.infoserver.servlet.QNServletResource*

```
public class QNServletResource
```

This class is a singleton for a property resource bundle to be used by the Query Net Wizard. The resource bundle contains values that could potentially displayed to users. It also provides a direct, static method for safely getting String values without worrying about MIssingResourceExceptions ras; 2/29/99; modified by S. Levine on 9/9/99

Attributes

resourceBundle

Operations

getBundle()
getInt(String)
getString(String)
initResourceBundle()

Attribute Detail

resourceBundle

```
private static ResourceBundle resourceBundle = null
```

The resource bundle we keep for use by the package.

Operation Detail

getBundle

```
public static ResourceBundle getBundle()
```

Return the single, static instance of the resource bundle for the SMPP Module

getInt

```
public static int getInt(String key)
```

Pass through to the resourceBundle, except that we internally catch the MissingResourceException and report the error but return a "?".
Return Int value.

getString

```
public static String getString(String key)
```

Pass through to the resourceBundle, except that we internally catch the MissingResourceException and report the error but return a "?"

initResourceBundle

```
private static void initResourceBundle()
```

Create the resource bundle

Class *tcs.ain.infoserver.servlet.QNWebPagePrinter*

```
public class QNWebPagePrinter
```

Extends:

tcs.web.WebPagePrinter

Constructors

QNWebPagePrinter(tcs.web.WebPage)

Operations

print()

Constructor Detail

QNWebPagePrinter

```
public QNWebPagePrinter(tcs.web.WebPage)
```

Operation Detail

print

```
public void print()
```

Class *tcs.ain.infoserver.servlet.QNWebPagePrinter*

```
public class QNWebPagePrinter
```

Extends:

tcs.web.WebPagePrinter

Base class for web page printing operations. Provides methods to print the header, banner, etc. Subclasses can provide distinct color / font / banner schemes for individual projects or pages

Constructors

QNWebPagePrinter(WebPage)

Operations

print()

Constructor Detail

QNWebPagePrinter

```
public QNWebPagePrinter(WebPage page)
```

Construct the page printer object. This assings the owner and gets the output stream

Operation Detail

print

```
public void print()
```

public print method; what client calls to start the printing process. This calls the basic components of the page (header, banner, body, footer)

Class *tcs.ain.infoserver.servlet.QueryNetResource*

```
class QueryNetResource
```

This class is a singleton for a property resource bundle to be used by the Query Net Wizard. The resource bundle contains values that could potentially displayed to users. It also provides a direct, static method for safely getting String values without worrying about MIssingResourceExceptions ras; 2/29/99; modified by S. Levine on 9/9/99

Attributes

resourceBundle

Operations

getBundle()
getString(String)
initResourceBundle()

Attribute Detail

resourceBundle

```
private static ResourceBundle resourceBundle = null
```

The resource bundle we keep for use by the package.

Operation Detail

getBundle

```
public static ResourceBundle getBundle()
```

Return the single, static instance of the resource bundle for the SMPP Module

getString

```
public static String getString(String key)
```

Pass through to the resourceBundle, except that we internally catch the MissingResourceException and report the error but return a "?"

initResourceBundle

```
private static void initResourceBundle()
```

Create the resource bundle

Class *tcs.ain.infoserver.servlet.StatusForm*

```
public class StatusForm
```

Extends:

tcs.ain.infoserver.servlet.DataWorkerWizardForm

Define the Data Sources that are desired.

Attributes

HtmlPage
testId

Constructors

StatusForm(HttpServletRequest, IRemoteRepository, String)

Operations

getDestinations()
getForm()
getTagTable()
populateFormFromWorker()
populateWorkerFromForm()

Attribute Detail

HtmlPage

```
protected static StringBuffer HtmlPage
```

Stores Html Page associated with a form producing class

testId

```
protected String testId
```

Id to be used for script testing only

Constructor Detail

StatusForm

```
public StatusForm(HttpServletRequest request, IRemoteRepository access, String stepDesc)
```

Constructor

Operation Detail

getDestinations

```
protected String getDestinations()
```

Return destinations associated with dataworker in html tag IMPORTANT: 1.) ONLY FILTERS WILL HAVE CHILD LISTENERS 2.) CHILD LISTENERS CAN BE OTHER FILTERS OR DESTINATIONS 3.) CHILD LISTENERS WILL NOT HAVE SIBLINGS 4.) DESTINATIONS WILL NOT HAVE CHILD LISTENERS ex/ ChangeOnlyFilter | SearchFilter | | | SMS Dest. Email Dest. Chat Dest.

getForm

```
public String getForm()
```

Return the appropriate form for adding new workers or selecting existing workers

getTagTable

```
protected Hashtable getTagTable()
```

Create a table of tags and their values to be used in replacing tags found in static html pages. Used to add dynamic content to these pages.

populateFormFromWorker

```
protected void populateFormFromWorker()
```

Populate the form fields with values from the DataWorker

populateWorkerFromForm

```
public String populateWorkerFromForm()
```

Populate the DataWorker with data assigned to the form

Class *tcs.ain.infoserver.servlet.ViewServicesPage*

```
public class ViewServicesPage
```

Extends:
tcs.ain.infoserver.servlet.InfoSvrPage

Attributes
HtmlPage

Constructors
ViewServicesPage(HttpServletRequest, HttpServletResponse, UserAgent)

Operations
getBody()
getDestinations(DataWorker)
GetPageId()
getTagTable()
processFormData(HttpServletRequest)
showServiceList()

Attribute Detail

HtmlPage

```
protected static StringBuffer HtmlPage
```

Stores Html Page associated with a form producing class

Constructor Detail

ViewServicesPage

```
public ViewServicesPage(HttpServletRequest request, HttpServletResponse response, UserAgent agent)
```

Constructor calls super() and sets the title for this page

Operation Detail

getBody

```
public String getBody()
```

Return the page's body. For increased efficiency, this directly writes to the response output and does not actually return any data.

getDestinations

```
protected String getDestinations(DataWorker worker)
```

Return destinations associated with dataworker in html tag IMPORTANT: 1.) ONLY FILTERS WILL HAVE CHILD LISTENERS 2.) CHILD LISTENERS CAN BE OTHER FILTERS OR DESTINATIONS 3.) CHILD LISTENERS WILL NOT HAVE SIBLINGS 4.) DESTINATIONS WILL NOT HAVE CHILD LISTENERS ex/ ChangeOnlyFilter | SearchFilter | | | SMS Dest. Email Dest. Chat Dest.

GetPageId

```
public static int GetPageId()
```

getTagTable

```
protected Hashtable getTagTable()
```

Create a table of tags and their values to be used in replacing tags found in static html pages. Used to add dynamic content to these pages.

processFormData

```
public void processFormData(HttpServletRequest request)
```

Start and stop workers based on user input

showServiceList

```
protected String showServiceList()
```

Create a table and show a summary of each of the services (ie, Workers owned by the userAgent. The worker titles will also contain hyperlins for more information and editing choices

Class *tcs.ain.infoserver.SimpleUserDestination*

```
public class SimpleUserDestination
```

Extends:

tcs.ain.infoserver.UserDeliveryDestination

Just send the output to STDOUT.

Operations

cleanUp()
getSaveables()
processData(DataEvent)
setSaveables(Object[])

Operation Detail

cleanUp

```
public void cleanUp()
```

Clean Up Data

getSaveables

```
public Object[] getSaveables()
```

Return an array of all objects that should be persistent. primitive datatypes must be converted to their Object equivalent. These objects will come back in the same order when setSaveables is called.

processData

```
public void processData(DataEvent evt)
```

I have new data; Send the data string to the screen

Throws:

InfoServerException

setSaveables

```
public void setSaveables(Object[] oa)
```

receive an array of objects that have been saved. the order will correspond to that specified when getSaveables was called. Users should handle Null values in the array.

Class *tcs.ain.infoserver.simpleweb.CustomEmailSourceGenerator*

```
public class CustomEmailSourceGenerator
```

This class is used to generated a customer Email Source

Operations

createEmailQueryEngine(String, String, String)
isValidMailHost(String)

Operation Detail

createEmailQueryEngine

```
public NewEmailQueryEngine createEmailQueryEngine(String hostName, String userId, String password)
```

Based on the parameters received from the request, attempt to create a NewEmailQueryEngine object. Throw an Exception with description of the problem if there is one. TO DO: DNS Validation, as with servlet, Remove Hard-Coded names

Throws:

InvalidParameterException
java.net.UnknownHostException

isValidMailHost

```
protected boolean isValidMailHost(String hostname)
```

Do the DNS MX lookup to determine if this host is valid or not

Class *tcs.ain.infoserver.simpleweb.CustomWebSourceFormGenerator*

```
public class CustomWebSourceFormGenerator
```

This presents all of the Public Data Sources and provides relevant information for any UserAgents that reference the public sources

Attributes

returnJsp
SaveButtonId

Constructors

CustomWebSourceFormGenerator(String)

Operations

getFormForDataWorker(DataWorker)
getFormHtml(Hashtable, String, String)
getNewFormHtml(String, String)
getPageParserQueryEngine(DataWorker)
getPageParserQueryEngineFromForm(HttpServletRequest)

Attribute Detail

returnJsp

```
String returnJsp
```

SaveButtonId

```
public static String SaveButtonId = "save"
```

Constructor Detail

CustomWebSourceFormGenerator

```
public CustomWebSourceFormGenerator(String returnJsp)
```

Called from the JspBean when user wants forms representing workers

Operation Detail

getFormForDataWorker

```
public String getFormForDataWorker(DataWorker worker)
```

Show the data selection form, but based on the PageParserQE already contained by this worker

Throws:
Exception

getFormHtml

```
protected String getFormHtml(Hashtable currentSelectedItems, String href, String startText)
```

Generate the form for selecting data from the href. setup the field to checked if currentSelectedItems contains the field already

Throws:
Exception

getNewFormHtml

```
public String getNewFormHtml(String href, String startText)
```

show a new form for selecting content startText can be null or 0 length if not relevant includes a 'saveform' submit button at the end.

Throws:
Exception

getPageParserQueryEngine

```
static PageParserQueryEngine getPageParserQueryEngine(DataWorker worker)
```

Return a PageParserQueryEngine if associated with this data source. return null if none found

getPageParserQueryEngineFromForm

```
public PageParserQueryEngine getPageParserQueryEngineFromForm(HttpServletRequest request)
```

Create and return a PageParserQueryEngine based on the form variables.

Throws:
Exception

Class *tcs.ain.infoserver.simpleweb.CustomWebSourceFormGenerator*

```
public class CustomWebSourceFormGenerator
```

Extends:
java.lang.Object

Attributes

returnJsp
SaveButtonId

Constructors

CustomWebSourceFormGenerator(java.lang.String)

Operations

getFormForDataWorker(tcs.ain.infoserver.DataWorker)
getFormHtml(java.util.Hashtable, java.lang.String, java.lang.String)

getNewFormHtml(java.lang.String, java.lang.String)
getPageParserQueryEngine(tcs.ain.infoserver.DataWorker)
getPageParserQueryEngineFromForm(javax.servlet.http.HttpServletRequest)

Attribute Detail

returnJsp

```
java.lang.String returnJsp
```

SaveButtonId

```
public static java.lang.String SaveButtonId
```

Constructor Detail

CustomWebSourceFormGenerator

```
public CustomWebSourceFormGenerator(java.lang.String)
```

Operation Detail

getFormForDataWorker

```
public java.lang.String getFormForDataWorker(tcs.ain.infoserver.DataWorker)
```

getFormHtml

```
protected java.lang.String getFormHtml(java.util.Hashtable, java.lang.String, java.lang.String)
```

getNewFormHtml

```
public java.lang.String getNewFormHtml(java.lang.String, java.lang.String)
```

getPageParserQueryEngine

```
static tcs.ain.infoserver.PageParserQueryEngine
getPageParserQueryEngine(tcs.ain.infoserver.DataWorker)
```

getPageParserQueryEngineFromForm

```
public tcs.ain.infoserver.PageParserQueryEngine
getPageParserQueryEngineFromForm(javax.servlet.http.HttpServletRequest)
```

Class *tcs.ain.infoserver.simpleweb.CustomWebSourceFormGenerator*

```
public class CustomWebSourceFormGenerator
```

This presents all of the Public Data Sources and provides relevant information for any UserAgents that reference the public sources

Attributes

returnJsp
SaveButtonId

Constructors

CustomWebSourceFormGenerator(String)

Operations

getFormForDataWorker(DataWorker)
getFormHtml(Hashtable, String, String)
getNewFormHtml(String, String)
getPageParserQueryEngine(DataWorker)
getPageParserQueryEngineFromForm(HttpServletRequest)

Attribute Detail

returnJsp

```
String returnJsp
```

SaveButtonId

```
public static String SaveButtonId = "save"
```

Constructor Detail

CustomWebSourceFormGenerator

```
public CustomWebSourceFormGenerator(String returnJsp)
```

Called from the JspBean when user wants forms representing workers

Operation Detail

getFormForDataWorker

```
public String getFormForDataWorker(DataWorker worker)
```

Show the data selection form, but based on the PageParserQE already contained by this worker This is useful when/if we want to edit an existing Web QueryEngine

Throws:
Exception

getFormHtml

```
protected String getFormHtml(Hashtable currentSelectedItems, String href, String startText)
```

Generate the form for selecting data from the href. setup the field to checked if currentSelectedItems contains the field already TO DO: It would be really great to sort entries by confidence level!

Throws:
Exception

getNewFormHtml

```
public String getNewFormHtml(String href, String startText)
```

show a new form for selecting content startText can be null or 0 length if not relevant includes a 'saveform' submit button at the end.

Throws:
Exception

getPageParserQueryEngine

```
static PageParserQueryEngine getPageParserQueryEngine(DataWorker worker)
```

Return a PageParserQueryEngine if associated with this data source. return null if none found

getPageParserQueryEngineFromForm

```
public PageParserQueryEngine getPageParserQueryEngineFromForm(HttpServletRequest request)
```

Create and return a PageParserQueryEngine based on the form variables.

Throws:
Exception

Class *tcs.ain.infoserver.simpleweb.CustomWebSourceGenerator*

```
public class CustomWebSourceGenerator
```

This presents all of the Public Data Sources and provides relevant information for any UserAgents that reference the public sources

Attributes

returnJsp
SaveButtonId

Constructors

CustomWebSourceGenerator(String)

Operations

getConfidenceIcon(int)
getFormForDataWorker(DataWorker, String)
getFormHtml(Hashtable, String, String, String)
getNewFormHtml(String, String, String)
getPageParserQueryEngine(DataWorker)
getPageParserQueryEngineFromForm(HttpServletRequest)
isValidLabel(String)

Attribute Detail

returnJsp

```
String returnJsp
```

SaveButtonId

```
public static String SaveButtonId = "save"
```

Constructor Detail

CustomWebSourceGenerator

```
public CustomWebSourceGenerator(String returnJsp)
```

Called from the JspBean when user wants forms representing workers

Operation Detail

getConfidenceIcon

```
protected String getConfidenceIcon(int level)
```

Return a representation of the confidence, based on the number.

getFormForDataWorker

```
public String getFormForDataWorker(DataWorker worker, String tableHeader)
```

Show the data selection form, but based on the PageParserQE already contained by this worker This is useful when/if we want to edit an existing Web QueryEngine

Throws:
Exception

getFormHtml

```
protected String getFormHtml(Hashtable currentSelectedItems, String href, String startText, String tableHeader)
```

Generate the form for selecting data from the href. setup the field to checked if currentSelectedItems contains the field already throw an exception if the site can not be accessed TO DO: It would be really great to sort entries by confidence level!

Throws:
Exception

getNewFormHtml

```
public String getNewFormHtml(String href, String startText, String tableHeader)
```

show a new form for selecting content startText can be null or 0 length if not relevant includes a 'saveform' submit button at the end.

Throws:
Exception

getPageParserQueryEngine

```
protected PageParserQueryEngine getPageParserQueryEngine(DataWorker worker)
```

Return a PageParserQueryEngine if associated with this data source. return null if none found

getPageParserQueryEngineFromForm

```
public PageParserQueryEngine getPageParserQueryEngineFromForm(HttpServletRequest request)
```

Create and return a PageParserQueryEngine based on the form variables.

Throws:
InvalidParameterException
java.net.UnknownHostException

isValidLabel

```
protected boolean isValidLabel(String label)
```

Does this consist of a-z or A-Z?

Class *tcs.ain.infoserver.simpleweb.InvalidParameterException*

```
public class InvalidParameterException
```

Extends:
java.lang.Exception

This can be thrown whenever a JSP passes parameters to an object and the parameters are missing, invalid, etc.

Constructors

InvalidParameterException(String)

Constructor Detail

InvalidParameterException

```
public InvalidParameterException(String s)
```

Class *tcs.ain.infoserver.simpleweb.JspBean*

```
public class JspBean
```

This provides some methods to simplify access to QNet and also return some dynamic HTML content.

Attributes

infoServerAccess
jdbcConn
UserAgentId
UserDataId
userDb

Constructors

JspBean()

Operations

createEmailQueryEngine(HttpServletRequest)
getAgentWorkerEditForms(HttpSession, String)
getCreatePublicWorkerForm(String, String)
getUserAgent(HttpSession)
getUserData(HttpSession)
getWorkerCount(HttpSession)
isSystemAvailable()
isValidSession(HttpSession)

login(String, String, HttpSession)
logout(HttpSession)
main(String[])
makeDbConnection()
makeRMIConnection()
processCustomWorkerCreateRequest(HttpSession, HttpServletRequest, QueryEngine)
processPublicWorkerCreateRequest(HttpSession, HttpServletRequest)
processWorkerChangeRequest(HttpSession, HttpServletRequest)
validateDbConnection()
validateRMIConnection()

Attribute Detail

infoServerAccess

```
protected IRemoteRepository infoServerAccess = null
```

jdbcConn

```
protected Connection jdbcConn = null
```

UserAgentId

```
static final String UserAgentId = "UserAgent"
```

UserDataId

```
static final String UserDataId = "UserData"
```

userDb

```
protected SMSAppDB userDb = null
```

Constructor Detail

JspBean

```
public JspBean()
```

Operation Detail

createEmailQueryEngine

```
public NewEmailQueryEngine createEmailQueryEngine(HttpServletRequest request)
```

Based on the parameters received from the request, attempt to create a NewEmailQueryEngine object. Throw an Exception with description of the problem if there is one. TO DO: DNS Validation, as with servlet, Remove Hard-Coded names

Throws:
Exception

getAgentWorkerEditForms

```
public String[] getAgentWorkerEditForms(HttpSession sess, String actionAddr)
```

Return array of HTML forms that presents all of the data workers for the user. The forms are populated with values assigned to the worker, and allow the values to be updated. Also includes forms for disabling, enabling, and remove public agents. returns null if none found

Parameter doc:
session,response, and the address of the 'page' that processes the form

getCreatePublicWorkerForm

```
public String getCreatePublicWorkerForm(String callbackPage, String buttonLabel)
```

Get the Form that is used to create a new Public Data Worker

getUserAgent

```
public UserAgent getUserAgent(HttpSession session)
```

Get the UserAgent object for this session could return null if not defined

getUserData

```
public UserData getUserData(HttpSession session)
```

Get the UserData object for this session could return null if not defined

getWorkerCount

```
public int getWorkerCount(HttpSession session)
```

Return how many worker the user in this session has

isSystemAvailable

```
public boolean isSystemAvailable()
```

Validate that the RMI and DB Connections are OK. Return true if all is OK.

isValidSession

```
public boolean isValidSession(HttpSession sess)
```

Return true if this is a valid session already; i.e., has a UserAgent and UserData for the logged in user

login

```
public boolean login(String userName, String password, HttpSession sess)
```

Is this a valid username & password? If so, a session object is created and the UserAgent and UserData are loaded into the session If the session already exists for userName/password, then return true; Note that if internal connection (RMI, DB) are bad, then this returns false with no indication.

logout

```
public void logout(HttpSession sess)
```

clear out the user session.

main

```
public static void main(String[] args)
```

makeDbConnection

```
protected void makeDbConnection()
```

Make sure jdbcConn is valid. If not, report the error to out and return false

Throws:

SQLException

makeRMIConnection

```
protected void makeRMIConnection()
```

Connect the remote object to the server if necessary

Throws:

Exception

processCustomWorkerCreateRequest

```
public void processCustomWorkerCreateRequest(HttpSession sess, HttpServletRequest request,
QueryEngine qe)
```

Process the request to create a worker from the specified data source TO DO: Have this created based on admin-assignable default values

processPublicWorkerCreateRequest

```
public void processPublicWorkerCreateRequest(HttpSession sess, HttpServletRequest request)
```

Process the request to create a worker from public data source This actually just gets the public QE and then calls processCustomWorkerCreateRequest

processWorkerChangeRequest

```
public boolean processWorkerChangeRequest(HttpSession sess, HttpServletRequest request)
```

Process the worker change request that would have originated from a form generated by getAgentWorkerEditForms() return the success of the operation

validateDbConnection

```
protected void validateDbConnection()
```

Make sure con is valid. If not, report the error to out and throw the exception TO DO: RECONNECT ON FAILURE

Throws:

SQLException

validateRMIConnection

```
protected boolean validateRMIConnection()
```

make connection if necessary; test the connection; if failed, try to to remake the connection return true if good connection

Class *tcs.ain.infoserver.simpleweb.JspBean*

```
public class JspBean
```

This provides some methods to simplify access to QNet and also return some dynamic HTML content.

Attributes

ChangeWorkerButtonId
CreatePublicWorkerButtonId
dbCheckSql
DeleteWorkerButtonId
infoServerAccess
jdbcConn
UserAgentId
UserDataId
userDb

Constructors

JspBean()

Operations

getAgentWorkerEditForms(HttpSession, String)
getCreatePublicWorkerForm(String, String)
getUserAgent(HttpSession)
getUserData(HttpSession)
getWorkerCount(HttpSession)
isSystemAvailable()
isValidSession(HttpSession)
login(String, String, HttpSession)
logout(HttpSession)
main(String[])
makeDbConnection()
makeRMIConnection()
processCustomWorkerCreateRequest(HttpSession, HttpServletRequest, QueryEngine)
processPublicWorkerCreateRequest(HttpSession, HttpServletRequest)
processWorkerChangeRequest(HttpSession, HttpServletRequest)
ShouldWorkerQueryAlways(String)
validateDbConnection()
validateRMIConnection()

Attribute Detail

ChangeWorkerButtonId

```
public static final String ChangeWorkerButtonId = "change"
```

CreatePublicWorkerButtonId

```
public static final String CreatePublicWorkerButtonId = "create"
```

dbCheckSql

```
protected String dbCheckSql = "select 'x' from dual"
```

DeleteWorkerButtonId

```
public static final String DeleteWorkerButtonId = "delete"
```

infoServerAccess

```
protected IRemoteRepository infoServerAccess = null
```

jdbcConn

```
protected Connection jdbcConn = null
```

UserAgentId

```
static final String UserAgentId = "UserAgent"
```

UserDataId

```
static final String UserDataId = "UserData"
```

userDb

```
protected SMSAppDB userDb = null
```

Constructor Detail

JspBean

```
public JspBean()
```

Operation Detail

getAgentWorkerEditForms

```
public String[] getAgentWorkerEditForms(HttpSession sess, String actionAddr)
```

Return array of HTML forms that presents all of the data workers for the user. The forms are populated with values assigned to the worker, and allow the values to be updated. Also includes forms for disabling, enabling, and remove public agents. returns null if none found

Parameter doc:

session,response, and the address of the 'page' that processes the form

getCreatePublicWorkerForm

```
public String getCreatePublicWorkerForm(String callbackPage, String buttonLabel)
```

Get the drop-down public list Form that is used to create a new Public Data Worker this is performed in here because of the complexity of the queries

getUserAgent

```
protected UserAgent getUserAgent(HttpSession session)
```

Get the UserAgent object for this session could return null if not defined protected for now; if JSP Really needs access, we can open it up

getUserData

```
public UserData getUserData(HttpSession session)
```

Get the UserData object for this session could return null if not defined

getWorkerCount

```
public int getWorkerCount(HttpSession session)
```

Return how many worker the user in this session has

isSystemAvailable

```
public boolean isSystemAvailable()
```

Validate that the RMI and DB Connections are OK. Return true if all is OK.

isValidSession

```
public boolean isValidSession(HttpSession sess)
```

Return true if this is a valid session already; i.e., has a UserAgent and UserData for the logged in user

login

```
public boolean login(String userName, String password, HttpSession sess)
```

Is this a valid username & password? If so, a session object is created and the UserAgent and UserData are loaded into the session If the session already exists for userName/password, then return true; Note that if internal connection (RMI, DB) are bad, then this returns false with no indication.

logout

```
public void logout(HttpSession sess)
```

clear out the user session.

main

```
public static void main(String[] args)
```

makeDbConnection

```
protected void makeDbConnection()
```

Make sure jdbcConn is valid. If not, report the error to out and return false

Throws:

SQLException

makeRMIConnection

```
protected void makeRMIConnection()
```

Connect the remote object to the server if necessary

Throws:

Exception

processCustomWorkerCreateRequest

```
public void processCustomWorkerCreateRequest(HttpSession sess, HttpServletRequest request,
QueryEngine qe)
```

Process the request to create a worker from the specified data source TO DO: Have this created based on admin-assignable default values

processPublicWorkerCreateRequest

```
public void processPublicWorkerCreateRequest(HttpSession sess, HttpServletRequest request)
```

Process the request to create a worker from public data source This actually just gets the public QE and then calls processCustomWorkerCreateRequest

processWorkerChangeRequest

```
public boolean processWorkerChangeRequest(HttpSession sess, HttpServletRequest request)
```

Process the worker change request that would have originated from a form generated by getAgentWorkerEditForms() return the success of the operation

ShouldWorkerQueryAlways

```
public static boolean ShouldWorkerQueryAlways(String qeId)
```

Some QueryEngines should be always queried; this determines if ID is one of them, as defined by queryAlways in the prop file

validateDbConnection

```
protected void validateDbConnection()
```

Make sure con is valid. If not, report the error to out and throw the exception

Throws:
SQLException

validateRMIConnection

```
protected boolean validateRMIConnection()
```

make connection if necessary; test the connection; if failed, try to to remake the connection return true if good connection

Class *tcs.ain.infoserver.simpleweb.JspBean*

```
public class JspBean
```

This provides some methods to simplify access to QNet and also return some dynamic HTML content.

Attributes

infoServerAccess
jdbcConn
UserAgentId
UserDataId
userDb

Constructors

JspBean()

Operations

createEmailQueryEngine(HttpServletRequest)
getAgentWorkerEditForms(HttpSession, String)
getCreatePublicWorkerForm(String)
getUserAgent(HttpSession)
getUserData(HttpSession)
getWorkerCount(HttpSession)
isSystemAvailable()
isValidSession(HttpSession)
login(String, String, HttpSession)
logout(HttpSession)
main(String[])
makeDbConnection()
makeRMIConnection()
processCustomWorkerCreateRequest(HttpSession, HttpServletRequest, QueryEngine)
processPublicWorkerCreateRequest(HttpSession, HttpServletRequest)
processWorkerChangeRequest(HttpSession, HttpServletRequest)
validateDbConnection()
validateRMIConnection()

Attribute Detail

infoServerAccess

```
protected IRemoteRepository infoServerAccess = null
```

jdbcConn

```
protected Connection jdbcConn = null
```

UserAgentId

```
static final String UserAgentId = "UserAgent"
```

UserDataId

```
static final String UserDataId = "UserData"
```

userDb

```
protected SMSAppDB userDb = null
```

Constructor Detail

JspBean

```
public JspBean()
```

Operation Detail

createEmailQueryEngine

```
public NewEmailQueryEngine createEmailQueryEngine(HttpServletRequest request)
```

Based on the parameters received from the request, attempt to create a NewEmailQueryEngine object. Throw an Exception with description of the problem if there is one. TO DO: DNS Validation, as with servlet

Throws:

Exception

getAgentWorkerEditForms

```
public String[] getAgentWorkerEditForms(HttpSession sess, String actionAddr)
```

Return array of HTML forms that presents all of the data workers for the user. The forms are populated with values assigned to the worker, and allow the values to be updated. Also includes forms for disabling, enabling, and remove public agents. returns null if none found

Parameter doc:

session,response, and the address of the 'page' that processes the form

getCreatePublicWorkerForm

```
public String getCreatePublicWorkerForm(String callbackPage)
```

Get the Form that is used to create a new Public Data Worker

getUserAgent

```
public UserAgent getUserAgent(HttpSession session)
```

Get the UserAgent object for this session could return null if not defined

getUserData

```
public UserData getUserData(HttpSession session)
```

Get the UserData object for this session could return null if not defined

getWorkerCount

```
public int getWorkerCount(HttpSession session)
```

Return how many worker the user in this session has

isSystemAvailable

```
public boolean isSystemAvailable()
```

Validate that the RMI and DB Connections are OK. Return true if all is OK.

isValidSession

```
public boolean isValidSession(HttpSession sess)
```

Return true if this is a valid session already; i.e., has a UserAgent and UserData for the logged in user

login

```
public boolean login(String userName, String password, HttpSession sess)
```

Is this a valid username & password? If so, a session object is created and the UserAgent and UserData are loaded into the session If the session already exists for userName/password, then return true; Note that if internal connection (RMI, DB) are bad, then this returns false with no indication.

logout

```
public void logout(HttpSession sess)
```

clear out the user session.

main

```
public static void main(String[] args)
```

makeDbConnection

```
protected void makeDbConnection()
```

Make sure jdbcConn is valid. If not, report the error to out and return false

Throws:

SQLException

makeRMIConnection

```
protected void makeRMIConnection()
```

Connect the remote object to the server if necessary

Throws:

Exception

processCustomWorkerCreateRequest

```
public void processCustomWorkerCreateRequest(HttpSession sess, HttpServletRequest request,
QueryEngine qe)
```

Process the request to create a worker from the specified data source TO DO: Have this created based on admin-assignable default values

processPublicWorkerCreateRequest

```
public void processPublicWorkerCreateRequest(HttpSession sess, HttpServletRequest request)
```

Process the request to create a worker from public data source This actually just gets the public QE and then calls processCustomWorkerCreateRequest

processWorkerChangeRequest

```
public boolean processWorkerChangeRequest(HttpSession sess, HttpServletRequest request)
```

Process the worker change request that would have originated from a form generated by getAgentWorkerEditForms() return the success of the operation

validateDbConnection

```
protected void validateDbConnection()
```

Make sure con is valid. If not, report the error to out and throw the exception TO DO: RECONNECT ON FAILURE

Throws:
SQLException

validateRMIConnection

```
protected void validateRMIConnection()
```

make connection if necessary; test the connection; if failed, try to to remake the connection return true if good connection

Class tcs.ain.infoserver.simpleweb.PublicAgentListGenerator

```
public class PublicAgentListGenerator
```

Extends:
java.lang.Object

Attributes

agentWorkers
infoServerAccess
userAgent
userData

Constructors

PublicAgentListGenerator(tcs.ain.infoserver.UserAgent, tcs.ain.appdata.UserData, tcs.ain.infoserver.access.IRemoteRepository)

Operations

getForms()

Attribute Detail

agentWorkers

```
protected tcs.ain.infoserver.DataWorker[] agentWorkers
```

infoServerAccess

```
protected tcs.ain.infoserver.access.IRemoteRepository infoServerAccess
```

userAgent

```
protected tcs.ain.infoserver.UserAgent userAgent
```

userData

```
protected tcs.ain.appdata.UserData userData
```

Constructor Detail

PublicAgentListGenerator

```
public PublicAgentListGenerator(tcs.ain.infoserver.UserAgent, tcs.ain.appdata.UserData,
tcs.ain.infoserver.access.IRemoteRepository)
```

Operation Detail

getForms

```
public java.lang.String[] getForms()
```

Class tcs.ain.infoserver.simpleweb.PublicAgentListGenerator

```
public class PublicAgentListGenerator
```

Extends:

java.lang.Object

Attributes

agentWorkers
infoServerAccess
userAgent
userData

Constructors

PublicAgentListGenerator(tcs.ain.infoserver.UserAgent, tcs.ain.appdata.UserData, tcs.ain.infoserver.access.IRemoteRepository)

Operations

getForms()

Attribute Detail

agentWorkers

```
protected tcs.ain.infoserver.DataWorker[] agentWorkers
```

infoServerAccess

```
protected tcs.ain.infoserver.access.IRemoteRepository infoServerAccess
```

userAgent

```
protected tcs.ain.infoserver.UserAgent userAgent
```

userData

```
protected tcs.ain.appdata.UserData userData
```

Constructor Detail

PublicAgentListGenerator

```
public PublicAgentListGenerator(tcs.ain.infoserver.UserAgent, tcs.ain.appdata.UserData,
tcs.ain.infoserver.access.IRemoteRepository)
```

Operation Detail

getForms

```
public java.lang.String[] getForms()
```

Class *tcs.ain.infoserver.simpleweb.PublicAgentListGenerator*

```
public class PublicAgentListGenerator
```

Extends:

java.lang.Object

Attributes

agentWorkers
infoServerAccess
userAgent
userData

Constructors

PublicAgentListGenerator(tcs.ain.infoserver.UserAgent, tcs.ain.appdata.UserData, tcs.ain.infoserver.access.IRemoteRepository)

Operations

getForms()

Attribute Detail

agentWorkers

```
protected tcs.ain.infoserver.DataWorker[] agentWorkers
```

infoServerAccess

```
protected tcs.ain.infoserver.access.IRemoteRepository infoServerAccess
```

userAgent

```
protected tcs.ain.infoserver.UserAgent userAgent
```

userData

```
protected tcs.ain.appdata.UserData userData
```

Constructor Detail

PublicAgentListGenerator

```
public PublicAgentListGenerator(tcs.ain.infoserver.UserAgent, tcs.ain.appdata.UserData,
tcs.ain.infoserver.access.IRemoteRepository)
```

Operation Detail

getForms

```
public java.lang.String[] getForms()
```

Class tcs.ain.infoserver.simpleweb.UserWorkFormGenerator

```
public class UserWorkFormGenerator
```

This presents all of the Public Data Sources and provides relevant information for any UserAgents that reference the public sources TO DO: Remove hard-coded English in the form use more constants for field ids

Attributes

agentWorkers
infoServerAccess
IsRunningFld
userAgent
userData
WorkerFld

Constructors

UserWorkFormGenerator(UserAgent, UserData, IRemoteRepository)

Operations

DoesSourceHaveEmailListener(IDataEventSource)
DoesSourceHaveSmsListener(IDataEventSource)
getChangeOnlyField(DataWorker)
getDestinationfields(DataWorker)
getForms(String)
getIsRunningField(DataWorker)
getTimerFields(DataWorker)

Attribute Detail

agentWorkers

```
protected DataWorker[] agentWorkers = null
```

infoServerAccess

```
protected IRemoteRepository infoServerAccess = null
```

IsRunningFld

```
public static final String IsRunningFld = "isrunning"
```

userAgent

```
protected UserAgent userAgent
```

userData

```
protected UserData userData
```

WorkerFld

```
public static final String WorkerFld = "workerid"
```

Constructor Detail

UserWorkFormGenerator

```
public UserWorkFormGenerator(UserAgent ua, UserData ud, IRemoteRepository ir)
```

Called from the JspBean when user wants forms representing workers

Operation Detail

DoesSourceHaveEmailListener

```
protected static boolean DoesSourceHaveEmailListener(IDataEventSource source)
```

Return whether this IDataEventSource has a listener of this class type (Email) Enumerates over the various listeners; recursively calls itself for Filter types

DoesSourceHaveSmsListener

```
static boolean DoesSourceHaveSmsListener(IDataEventSource source)
```

Return whether this IDataEventSource has a listener of this class type (SMS) Enumerates over the various listeners; recursively calls itself for Filter types There must be a more efficient way to do this

getChangeOnlyField

```
protected StringBuffer getChangeOnlyField(DataWorker worker)
```

return the ChangeOnly label and checkbox

getDestinationfields

```
protected StringBuffer getDestinationfields(DataWorker worker)
```

Return fields for whether SMS and/or Email Destinations are to be used

getForms

```
public String[] getForms(String actionAddr)
```

Return an array of Strings, each of which contains a form for viewing and editing the worker agents. An array is returned because the calling JSP page might want to put each form in a table cell, or something similar for formatting purposes. receive page address that will process the form action

getIsRunningField

```
protected StringBuffer getIsRunningField(DataWorker worker)
```

Get fields for whether agent is running

getTimerFields

```
protected StringBuffer getTimerFields(DataWorker worker)
```

Return the timer selection fields for the queryengine Min of hour, Hour of day, DOW, DOM

Class *tcs.ain.infoserver.simpleweb.UserWorkFormGenerator*

```
public class UserWorkFormGenerator
```

This presents all of the Public Data Sources and provides relevant information for any UserAgents that reference the public sources

Attributes

agentWorkers
EveryIdLbl
infoServerAccess
IsRunningFld
userAgent
userData
WorkerFld

Constructors

UserWorkFormGenerator(UserAgent, UserData, IRemoteRepository)

Operations

DoesSourceHaveEmailListener(IDataEventSource)
DoesSourceHaveSmsListener(IDataEventSource)
getChangeOnlyField(DataWorker)
getDestinationfields(DataWorker)
getForms(String)
getIsRunningField(DataWorker)
getTimerFields(DataWorker)

Attribute Detail

agentWorkers

```
protected DataWorker[] agentWorkers = null
```

EveryIdLbl

```
public static String EveryIdLbl = "Every"
```

infoServerAccess

```
protected IRemoteRepository infoServerAccess = null
```

IsRunningFld

```
public static final String IsRunningFld = "isrunning"
```

userAgent

```
protected UserAgent userAgent
```

userData

```
protected UserData userData
```

WorkerFld

```
public static final String WorkerFld = "workerid"
```

Constructor Detail

UserWorkFormGenerator

```
public UserWorkFormGenerator(UserAgent ua, UserData ud, IRemoteRepository ir)
```

Called from the JspBean when user wants forms representing workers

Operation Detail

DoesSourceHaveEmailListener

```
protected static boolean DoesSourceHaveEmailListener(IDataEventSource source)
```

Return whether this IDataEventSource has a listener of this class type (Email) Enumerates over the various listeners; recursively calls itself for Filter types

DoesSourceHaveSmsListener

```
static boolean DoesSourceHaveSmsListener(IDataEventSource source)
```

Return whether this IDataEventSource has a listener of this class type (SMS) Enumerates over the various listeners; recursively calls itself for Filter types There must be a more efficient way to do this

getChangeOnlyField

```
protected StringBuffer getChangeOnlyField(DataWorker worker)
```

return the ChangeOnly label and checkbox

getDestinationfields

```
protected StringBuffer getDestinationfields(DataWorker worker)
```

Return fields for whether SMS and/or Email Destinations are to be used

getForms

```
public String[] getForms(String actionAddr)
```

Return an array of Strings, each of which contains a form for viewing and editing the worker agents. An array is returned because the calling JSP page might want to put each form in a table cell, or something similar for formatting purposes. receive page address that will process the form action

getIsRunningField

```
protected StringBuffer getIsRunningField(DataWorker worker)
```

Get fields for whether agent is running

getTimerFields

```
protected StringBuffer getTimerFields(DataWorker worker)
```

Return the timer selection fields for the queryengine Min of hour, Hour of day, DOW, DOM

Class *tcs.ain.infoserver.simpleweb.UserWorkFormGenerator*

```
public class UserWorkFormGenerator
```

This presents all of the Public Data Sources and provides relevant information for any UserAgents that reference the public sources

Attributes

agentWorkers
EveryIdLbl
infoServerAccess
IsRunningFld
userAgent
userData
WorkerFld

Constructors

UserWorkFormGenerator(UserAgent, UserData, IRemoteRepository)

Operations

DoesSourceHaveEmailListener(IDataEventSource)
DoesSourceHaveSmsListener(IDataEventSource)
getChangeOnlyField(DataWorker)
getDestinationfields(DataWorker)
getForms(String)

getIsRunningField(DataWorker)
getTimerFields(DataWorker)

Attribute Detail

agentWorkers

```
protected DataWorker[] agentWorkers = null
```

EveryIdLbl

```
public static String EveryIdLbl = "Every"
```

infoServerAccess

```
protected IRemoteRepository infoServerAccess = null
```

IsRunningFld

```
public static final String IsRunningFld = "isrunning"
```

userAgent

```
protected UserAgent userAgent
```

userData

```
protected UserData userData
```

WorkerFld

```
public static final String WorkerFld = "workerid"
```

Constructor Detail

UserWorkFormGenerator

```
public UserWorkFormGenerator(UserAgent ua, UserData ud, IRemoteRepository ir)
```

Called from the JspBean when user wants forms representing workers

Operation Detail

DoesSourceHaveEmailListener

```
protected static boolean DoesSourceHaveEmailListener(IDataEventSource source)
```

Return whether this IDataEventSource has a listener of this class type (Email) Enumerates over the various listeners; recursively calls itself for Filter types

DoesSourceHaveSmsListener

```
static boolean DoesSourceHaveSmsListener(IDataEventSource source)
```

Return whether this IDataEventSource has a listener of this class type (SMS) Enumerates over the various listeners; recursively calls itself for Filter types There must be a more efficient way to do this

getChangeOnlyField

```
protected StringBuffer getChangeOnlyField(DataWorker worker)
```

return the ChangeOnly label and checkbox

getDestinationfields

```
protected StringBuffer getDestinationfields(DataWorker worker)
```

Return fields for whether SMS and/or Email Destinations are to be used

getForms

```
public String[] getForms(String actionAddr)
```

Return an array of Strings, each of which contains a form for viewing and editing the worker agents. An array is returned because the calling JSP page might want to put each form in a table cell, or something similar for formatting purposes. receive page address that will process the form action

getIsRunningField

```
protected StringBuffer getIsRunningField(DataWorker worker)
```

Get fields for whether agent is running

getTimerFields

```
protected StringBuffer getTimerFields(DataWorker worker)
```

Return the timer selection fields for the queryengine Min of hour, Hour of day, DOW, DOM

Class *tcs.ain.infoserver.simpleweb.UserWorkFormProcessor*

```
public class UserWorkFormProcessor
```

This guy applies change requests for a worker against the useragent, which is obtained by querying the session

Attributes

- dataWorker
- infoServerAccess
- request
- userAgent
- userData

Constructors

- UserWorkFormProcessor(UserAgent, UserData, IRemoteRepository, HttpServletRequest)

Operations

- addEmailSupport(IDataSource, IDataEventSource)
- addSmsSupport(IDataSource, IDataEventSource)
- applyChangeOnly()
- applyDestinationFields()
- applyIsRunning()
- applyTimerFields()
- getWorkerChangeOnlyFilter()
- processForm()
- removeEmailSupport(IDataEventSource)
- removeSmsSupport(IDataEventSource)

Attribute Detail

dataWorker

```
protected DataWorker dataWorker = null
```

infoServerAccess

```
protected IRemoteRepository infoServerAccess = null
```

request

```
HttpServletRequest request = null
```

userAgent

```
protected UserAgent userAgent = null
```

userData

```
protected UserData userData = null
```

Constructor Detail

UserWorkFormProcessor

```
public UserWorkFormProcessor(UserAgent ua, UserData ud, IRemoteRepository ir, HttpServletRequest request)
```

Called from the JspBean when we need to apply the results of having clicked on a form generated bby the UserWorkFormGenerator

Operation Detail

addEmailSupport

```
protected void addEmailSupport(IDataSource dataSource, IDataEventSource evtSource)
```

add Email listener to this source TO DO: Pull Email constants from property file

addSmsSupport

```
protected void addSmsSupport(IDataSource dataSource, IDataEventSource evtSource)
```

add Email listener to this source TO DO: Pull SMS constants from property file

applyChangeOnly

```
protected void applyChangeOnly()
```

Add or Remote a ChangeOnlyFilter, as directed by the checkbox.

Throws:
Exception

applyDestinationFields

```
protected void applyDestinationFields()
```

Apply the request to add/remove Destination TO DO: Make sure that what's happening with applyChangeOnly does not strangely affect this and vice-versa

applyIsRunning

```
protected void applyIsRunning()
```

Check and apply the isRunning field

Throws:
Exception

applyTimerFields

```
protected void applyTimerFields()
```

apply the timer fields; taking into special consideration the 'combined' time possibilities

Throws:
Exception

getWorkerChangeOnlyFilter

```
protected ChangeOnlyFilter getWorkerChangeOnlyFilter()
```

Return a ChangeOnlyFilter that may be assigned to the dataWorker. If none is assigned, return null

processForm

```
public boolean processForm()
```

Check each of the possible fields and then save the agent return false if the form could not be processed; Note: We could also throw an exception to give more detail;

removeEmailSupport

```
protected void removeEmailSupport(IDataEventSource source)
```

Remove EmailDestination from the DataEventSource

removeSmsSupport

```
protected void removeSmsSupport(IDataEventSource source)
```

Remove SmsDestination from the DataEventSource

Class *tcs.ain.infoserver.simpleweb.UserWorkFormProcessor*

```
public class UserWorkFormProcessor
```

This guy applies change requests for a worker against the useragent, which is obtained by querying the session

Attributes

dataWorker
infoServerAccess
request
userAgent
userData

Constructors

UserWorkFormProcessor(UserAgent, UserData, IRemoteRepository, HttpServletRequest)

Operations

addEmailSupport(IDataSource, IDataEventSource)
addSmsSupport(IDataSource, IDataEventSource)
applyChangeOnly()
applyDestinationFields()
applyIsRunning()
applyTimerFields()
getWorkerChangeOnlyFilter()
processForm()
removeEmailSupport(IDataEventSource)
removeSmsSupport(IDataEventSource)

Attribute Detail

dataWorker

```
protected DataWorker dataWorker = null
```

infoServerAccess

```
protected IRemoteRepository infoServerAccess = null
```

request

```
HttpServletRequest request = null
```

userAgent

```
protected UserAgent userAgent = null
```

userData

```
protected UserData userData = null
```

Constructor Detail

UserWorkFormProcessor

```
public UserWorkFormProcessor(UserAgent ua, UserData ud, IRemoteRepository ir, HttpServletRequest
request)
```

Called from the JspBean when we need to apply the results of having clicked on a form generated bby the UserWorkFormGenerator

Operation Detail

addEmailSupport

```
protected void addEmailSupport(IDataSource dataSource, IDataEventSource evtSource)
```

add Email listener to this source TO DO: Pull Email constants from property file

addSmsSupport

```
protected void addSmsSupport(IDataSource dataSource, IDataEventSource evtSource)
```

add Email listener to this source TO DO: Pull SMS constants from property file

applyChangeOnly

```
protected void applyChangeOnly()
```

Add or Remote a ChangeOnlyFilter, as directed by the checkbox.

Throws:
Exception

applyDestinationFields

```
protected void applyDestinationFields()
```

Apply the request to add/remove Destination TO DO: Make sure that what's happening with applyChangeOnly does not strangely affect this and vice-versa

applyIsRunning

```
protected void applyIsRunning()
```

Check and apply the isRunning field

Throws:
Exception

applyTimerFields

```
protected void applyTimerFields()
```

apply the timer fields; taking into special consideration the 'combined' time possibilities

Throws:
Exception

getWorkerChangeOnlyFilter

```
protected ChangeOnlyFilter getWorkerChangeOnlyFilter()
```

Return a ChangeOnlyFilter that may be assigned to the dataWorker. If none is assigned, return null

processForm

```
public boolean processForm()
```

Check each of the possible fields and then save the agent return false if the form could not be processed; Note: We could also throw an exception to give more detail;

removeEmailSupport

```
protected void removeEmailSupport(IDataEventSource source)
```

Remove EmailDestination from the DataEventSource

removeSmsSupport

```
protected void removeSmsSupport(IDataEventSource source)
```

Remove SmsDestination from the DataEventSource

Class tcs.ain.infoserver.simpleweb.UserWorkFormProcessor

```
public class UserWorkFormProcessor
```

This guy applies change requests for a worker against the useragent, which is obtained by querying the session

Attributes

dataWorker
infoServerAccess
request
userAgent
userData

Constructors

UserWorkFormProcessor(UserAgent, UserData, IRemoteRepository, HttpServletRequest)

Operations

addEmailSupport(IDataSource, IDataEventSource)
addSmsSupport(IDataSource, IDataEventSource)
applyChangeOnly()
applyDestinationFields()
applyIsRunning()
applyTimerFields()
getWorkerChangeOnlyFilter()
processForm()
removeEmailSupport(IDataEventSource)
removeSmsSupport(IDataEventSource)

Attribute Detail

dataWorker

```
protected DataWorker dataWorker = null
```

infoServerAccess

```
protected IRemoteRepository infoServerAccess = null
```

request

```
HttpServletRequest request = null
```

userAgent

```
protected UserAgent userAgent = null
```

userData

```
protected UserData userData = null
```

Constructor Detail

UserWorkFormProcessor

```
public UserWorkFormProcessor(UserAgent ua, UserData ud, IRemoteRepository ir, HttpServletRequest
request)
```

Called from the JspBean when we need to apply the results of having clicked on a form generated bby the UserWorkFormGenerator

Operation Detail

addEmailSupport

```
protected void addEmailSupport(IDataSource dataSource, IDataEventSource evtSource)
```

add Email listener to this source TO DO: Pull Email constants from property file

addSmsSupport

```
protected void addSmsSupport(IDataSource dataSource, IDataEventSource evtSource)
```

add Email listener to this source TO DO: Pull SMS constants from property file

applyChangeOnly

```
protected void applyChangeOnly()
```

Add or Remote a ChangeOnlyFilter, as directed by the checkbox.

Throws:
Exception

applyDestinationFields

```
protected void applyDestinationFields()
```

Apply the request to add/remove Destination TO DO: Make sure that what's happening with applyChangeOnly does not strangely affect this and vice-versa

applyIsRunning

```
protected void applyIsRunning()
```

Check and apply the isRunning field

Throws:
Exception

applyTimerFields

```
protected void applyTimerFields()
```

apply the timer fields; taking into special consideration the 'combined' time possibilities

Throws:
Exception

getWorkerChangeOnlyFilter

```
protected ChangeOnlyFilter getWorkerChangeOnlyFilter()
```

Return a ChangeOnlyFilter that may be assigned to the dataWorker. If none is assigned, return null

processForm

```
public boolean processForm()
```

Check each of the possible fields and then save the agent return false if the form could not be processed; Note: We could also throw an exception to give more detail;

removeEmailSupport

```
protected void removeEmailSupport(IDataEventSource source)
```

Remove EmailDestination from the DataEventSource

removeSmsSupport

```
protected void removeSmsSupport(IDataEventSource source)
```

Remove SmsDestination from the DataEventSource

Class *tcs.ain.infoserver.SMSUserDestination*

```
public class SMSUserDestination
```

Extends:
tcs.ain.infoserver.UserDeliveryDestination

This sends its data via RMI to the Web Gateway's queue

Attributes

destMin
queue
remoteAddress

Constructors

SMSUserDestination()

Operations

cleanUp()
getDestMin()
getRemoteAddress()
getRMIconnected()
getSaveables()
makeRMIConnection()
processData(DataEvent)
sendShortMessage(String, String)
setDestMin(String)
setRemoteAddress(String)
setSaveables(Object[])

Attribute Detail

destMin

```
protected String destMin
```

The MIN where we will send the message

queue

```
protected static IQueueProxy queue = null
```

the remote queue object for inserting messages

remoteAddress

```
protected String remoteAddress = "localhost"
```

The address of the RemoteQueue

Constructor Detail

SMSUserDestination

```
public SMSUserDestination()
```

Operation Detail

cleanUp

```
public void cleanUp()
```

Clean up data

getDestMin

```
public String getDestMin()
```

getRemoteAddress

```
public String getRemoteAddress()
```

getRMIconnected

```
protected boolean getRMIconnected()
```

make connection if necessary; test the connection; if failed, try to to remake the connection return true if good connection

Throws:
Exception

getSaveables

```
public Object[] getSaveables()
```

Return an array of all objects that should be persistent. primitive datatypes must be converted to their Object equivalent. These objects will come back in the same order when setSaveables is called.

makeRMIConnection

```
protected void makeRMIConnection()
```

Connect the remote object to the server if necessary

Throws:
Exception

processData

```
public void processData(DataEvent evt)
```

I have new data; it to the gateway

Throws:
SMSUserDestinationException

sendShortMessage

```
protected void sendShortMessage(String address, String msgTxt)
```

Send a short message

Throws:
Exception

setDestMin

```
public void setDestMin(String destMin)
```

setRemoteAddress

```
public void setRemoteAddress(String addr)
```

setSaveables

```
public void setSaveables(Object[] oa)
```

receive an array of objects that have been saved. the order will correspond to that specified when getSaveables was called. Users should handle Null values in the array.

Class *tcs.ain.infoserver.SMSUserDestinationException*

```
public class SMSUserDestinationException
```

Extends:
tcs.ain.infoserver.InfoServerException

Constructors

SMSUserDestinationException(String)

Constructor Detail

SMSUserDestinationException

```
public SMSUserDestinationException(String s)
```

Class *tcs.ain.infoserver.tests.GSWDS*

```
public class GSWDS
```

Tests the IMAP Query Engine; query every one minute

Constructors

GSWDS()

Operations

main(String[])

Constructor Detail

GSWDS

```
public GSWDS()
```

Throws:

Exception

Operation Detail

main

```
public static void main(String[] args)
```

Class *tcs.ain.infoserver.tests.Test1*

```
public class Test1
```

This class create a WebQueryEngine, 2 DataWorkers, SimpleUserDestination and ties them all together. The dataworker will execute once, causing the XML from the horoscope web page to be seen on the screen.

Constructors

Test1()

Operations

main(String[])

Constructor Detail

Test1

```
public Test1()
```

Operation Detail

main

```
public static void main(String[] args)
```

Class *tcs.ain.infoserver.tests.Test10*

```
public class Test10
```

Adds on to Test3 by saving to the Repository.

Constructors

Test10()

Operations

main(String[])

Constructor Detail

Test10

```
public Test10()
```

Throws:

Exception

Operation Detail

main

```
public static void main(String[] args)
```

Class *tcs.ain.infoserver.tests.Test11*

```
public class Test11
```

Tests the IMAP Query Engine; query every one minute

Constructors

Test11()

Operations

main(String[])

Constructor Detail

Test11

```
public Test11()
```

Throws:

Exception

Operation Detail

main

```
public static void main(String[] args)
```

Class *tcs.ain.infoserver.tests.Test12*

```
public class Test12
```

This test just shows all of the Data Sources and interleaves userUser info If the user has 'signed up' with this source, then user info is shown.

Attributes

agentWorkers
rep
userAgent

Constructors

Test12(String)

Operations

main(String[])
showAllUserWorkers()

showInfo()
showUserMatch(String)

Attribute Detail

agentWorkers

```
DataWorker[] agentWorkers = null
```

rep

```
IRemoteRepository rep = null
```

userAgent

```
UserAgent userAgent = null
```

Constructor Detail

Test12

```
public Test12(String uName)
```

Operation Detail

main

```
public static void main(String[] args)
```

showAllUserWorkers

```
protected void showAllUserWorkers()
```

showInfo

```
protected void showInfo()
```

Throws:
Exception

showUserMatch

```
protected void showUserMatch(String sourceKey)
```

Class *tcs.ain.infoserver.tests.Test2*

```
public class Test2
```

Adds on to Test1 by assigned an XSL doc to the DataSource, and assigning a formatter that uses that XSL doc.

Constructors

Test2()

Operations

main(String[])

Constructor Detail

Test2

```
public Test2()
```

Throws:
Exception

Operation Detail

main

```
public static void main(String[] args)
```

Class tcs.ain.infoserver.tests.Test3

```
public class Test3
```

Adds on to Test2 by using the CalendarTimer.

Constructors

Test3()

Operations

main(String[])

Constructor Detail

Test3

```
public Test3()
```

Throws:

Exception

Operation Detail

main

```
public static void main(String[] args)
```

Class tcs.ain.infoserver.tests.Test4

```
public class Test4
```

Extends:

java.lang.Object

Constructors

Test4()

Operations

main(java.lang.String[])

Constructor Detail

Test4

```
public Test4()
```

Operation Detail

main

```
public static void main(java.lang.String[])
```

Class tcs.ain.infoserver.tests.Test5

```
public class Test5
```

Loads the objects saved in Test4. Workers will start to auto-run. This is what InfoServer.main does also.

Constructors

Test5()

Operations

main(String[])

Constructor Detail

Test5

```
public Test5()
```

Throws:

Exception

Operation Detail

main

```
public static void main(String[] args)
```

Class *tcs.ain.infoserver.tests.Test6*

```
public class Test6
```

Gets the InfoserverAccess object. Creates an UserAgent and saves it to the InfoServerAccess remote object.

Constructors

Test6()

Operations

main(String[])

Constructor Detail

Test6

```
public Test6()
```

Throws:

Exception

Operation Detail

main

```
public static void main(String[] args)
```

Class *tcs.ain.infoserver.tests.Test7*

```
public class Test7
```

Gets the InfoserverAccess object. Retrieves the UserAgent saved in Test6.

Constructors

Test7()

Operations

main(String[])

Constructor Detail

Test7

```
public Test7()
```

Throws:
Exception

Operation Detail

main

```
public static void main(String[] args)
```

Class tcs.ain.infoserver.tests.Test8

```
public class Test8
```

Tests the IMAP Query Engine; query every one minute

Constructors

Test8()

Operations

main(String[])

Constructor Detail

Test8

```
public Test8()
```

Throws:
Exception

Operation Detail

main

```
public static void main(String[] args)
```

Class tcs.ain.infoserver.tests.Test9

```
public class Test9
```

Single worker, single data source for IMAP, with 2 XSL styles. Two destinations, each using a different style sheet. Dest 1 inserts the Email data into a database for the IVR reader Dest 2 displays the at the MIN.

Constructors

Test9(String, String, String)

Operations

main(String[])

Constructor Detail

Test9

```
public Test9(String emailServer, String emailAcct, String emailPW)
```

Throws:
Exception

Operation Detail

main

```
public static void main(String[] args)
```

Class *tcs.ain.infoserver.tests.TestPageParserQE*

```
public class TestPageParserQE
```

This creates a PageParserQueryEngine and has a DataWorker query it

Constructors

TestPageParserQE()

Operations

main(String[])

Constructor Detail

TestPageParserQE

```
public TestPageParserQE()
```

Throws:
Exception

Operation Detail

main

```
public static void main(String[] args)
```

Class *tcs.ain.infoserver.tests.WLabDS*

```
public class WLabDS
```

Tests the IMAP Query Engine; query every one minute

Constructors

WLabDS()

Operations

main(String[])

Constructor Detail

WLabDS

```
public WLabDS()
```

Throws:
Exception

Operation Detail

main

```
public static void main(String[] args)
```

Class tcs.ain.infoserver.TimeElement

```
public class TimeElement
```

Implements:

java.io.Serializable
tcs.persist.TCSPersistent

This provides services for identifying time elements used by the CalendarTimer. Can represent minutes, hours, day of week, etc.

Attributes

timeUnit
timeValues

Constructors

TimeElement()

Operations

addValue(int)
doAlways()
getSaveables()
getTimeUnit()
getTimeValues()
getValues()
hasMatch(Date)
isDoAlways()
removeAllTimerValues()
setSaveables(Object[])
setTimeUnit(int)
setTimeValues(Vector)

Attribute Detail

timeUnit

```
private int timeUnit
```

Day of week, Day of Month, etc

timeValues

```
private Vector timeValues = new Vector()
```

The list of values that can be matched against a time in hasMatch for the assigned timeUnit

Constructor Detail

TimeElement

```
public TimeElement()
```

Operation Detail

addValue

```
public void addValue(int value)
```

Add a value that will indicate a valid time. When hasMatch(Date) is called, we check against values populated with this method using the current time unit.

doAlways

```
public void doAlways()
```

Set this timer to run at every interval (doAlways)

getSaveables

```
public Object[] getSaveables()
```

getTimeUnit

```
public int getTimeUnit()
```

tgGet

getTimeValues

```
public Vector getTimeValues()
```

tgGet

getValues

```
public int[] getValues()
```

Return an array of the values that have been assigned.

hasMatch

```
public boolean hasMatch(Date mdate)
```

Whether we have a time value that matches the received Date for the assigned timeUnit

isDoAlways

```
public boolean isDoAlways()
```

Whether we are assigned to always have a match for the time unit

removeAllTimerValues

```
public void removeAllTimerValues()
```

Remove all time values

setSaveables

```
public void setSaveables(Object[] oa)
```

setTimeUnit

```
public void setTimeUnit(int timeUnit)
```

tgSet

setTimeValues

```
public void setTimeValues(Vector v)
```

tgSet

Class *tcs.ain.infoserver.Timer*

```
public class Timer
```

Extends:

java.lang.Thread

Implements:

java.io.Serializable
tcs.persist.TCSPersistent

Used by Worker to control when the worker should do its work. Different types of timers can fire according to time elapsed, # of events, time of day, etc. This timer just fires once at start.

Attributes

worker

Constructors

Timer()

Operations

getSaveables()
getWorker()
run()
setSaveables(Object[])
setWorker(Worker)

Attribute Detail

worker

```
protected Worker worker
```

Constructor Detail

Timer

```
public Timer()
```

Operation Detail

getSaveables

```
public Object[] getSaveables()
```

Return an array of all objects that should be persistent. primitive datatypes must be converted to their Object equivalent. These objects will come back in the same order when setSaveables is called.

getWorker

```
public Worker getWorker()
```

run

```
public void run()
```

When the thread runs, tell the worker to do its work

setSaveables

```
public void setSaveables(Object[] oa)
```

receive an array of objects that have been saved. the order will correspond to that specified when getSaveables was called. Users should handle Null values in the array.

setWorker

```
public void setWorker(Worker worker)
```

Class *tcs.ain.infoserver.UserAgent*

```
public class UserAgent
```

Extends:

tcs.ain.infoserver.InfoServerElement

UserUser is a collection of information about the User. It also collects all of the InfoServerElements desgnated to this user. The user's workers, dataDestinations, private dataSources, etc. are contained by this user.

Attributes

email
fName
lName
mobileNumber
userElements

Constructors

UserAgent()

Operations

addElement(InfoServerElement)
getAllDataWorkers()
getAllElements()
getDataWorker(String)
getEmail()
getFName()
getLName()
getMobileNumber()
getSaveables()
getUserElements()
getWorkerIds()
removeElement(InfoServerElement)
setEmail(String)
setFName(String)
setLName(String)
setMobileNumber(String)
setSaveables(Object[])
setUserElements(Vector)
startWorkers()
stopWorkers()

Attribute Detail

email

```
private String email = new String()
```

fName

```
private String fName
```

lName

```
private String lName
```

mobileNumber

```
private String mobileNumber
```

userElements

```
private Vector userElements = new Vector()
```

Constructor Detail

UserAgent

```
public UserAgent()
```

Operation Detail

addElement

```
public void addElement(InfoServerElement e)
```

Add an InfoServer Element for the user

getAllDataWorkers

```
public DataWorker[] getAllDataWorkers()
```

Return an array of all the DataWorkers that I possess *

getAllElements

```
public InfoServerElement[] getAllElements()
```

Return an array of all the InfoServerElements I possess (This could be of various types*

getDataWorker

```
public DataWorker getDataWorker(String workerId)
```

getEmail

```
public String getEmail()
```

tgGet

getFName

```
public String getFName()
```

tgGet

getLName

```
public String getLName()
```

tgGet

getMobileNumber

```
public String getMobileNumber()
```

tgGet

getSaveables

```
public Object[] getSaveables()
```

Return an array of all objects that should be persistent. primitive datatypes must be converted to their Object equivalent. These objects will come back in the same order when setSaveables is called.

getUserElements

```
public Vector getUserElements()
```

tgGet

getWorkerIds

```
public Vector getWorkerIds()
```

Return an array of all the DataWorkers that I possess *

removeElement

```
public void removeElement(InfoServerElement e)
```

Remove an InfoServer Elements from the user

setEmail

```
public void setEmail(String email)
```

tgSet

setFName

```
public void setFName(String fName)
```

tgSet

setLName

```
public void setLName(String lName)
```

tgSet

setMobileNumber

```
public void setMobileNumber(String mobileNumber)
```

tgSet

setSaveables

```
public void setSaveables(Object[] oa)
```

receive an array of objects that have been saved. the order will correspond to that specified when getSaveables was called. Users should handle Null values in the array.

setUserElements

```
public void setUserElements(Vector userElements)
```

tgSet

startWorkers

```
public void startWorkers()
```

Start all the workers assigned to me if they should be running. Called when de-serialized by Repository.

stopWorkers

```
public void stopWorkers()
```

Stop all the workers assigned to me.

Class *tcs.ain.infoserver.UserDeliveryDestination*

```
public abstract class UserDeliveryDestination
```

Extends:

tcs.ain.infoserver.InfoServerElement

Implements:

tcs.ain.infoserver.IDataEventListener

This implementation uses a DataFormatter for retrieving the info from the DataSource and returning a String representation. Subclasses of this will implement the deliverData() method, which will take the string representation of the data and send it to the end-user in an appropriate manner.

Attributes

dataDestinations
formatter

Constructors

UserDeliveryDestination()

Operations

cleanUp()
getFormatter()
processData(DataEvent)
setFormatter(DataFormatter)

Attribute Detail

dataDestinations

```
protected Vector dataDestinations
```

Associates:
IDataEventListener

formatter

```
protected DataFormatter formatter = new DataFormatter()
```

Default Data Formatter

Constructor Detail

UserDeliveryDestination

```
public UserDeliveryDestination()
```

Operation Detail

cleanUp

```
public abstract void cleanUp()
```

Required method for a IDataEventListener. What tells me that I should examine a data source.

getFormatter

```
public DataFormatter getFormatter()
```

tgGet

processData

```
public abstract void processData(DataEvent evt)
```

Required method for a IDataEventListener. What tells me that I should examine a data source.

Throws:
InfoServerException

setFormatter

```
public void setFormatter(DataFormatter formatter)
```

tgSet

Class *tcs.ain.infoserver.WebQueryEngine*

```
public class WebQueryEngine
```

Extends:
tcs.ain.infoserver.JDBCQueryEngine

This handles querying a web page. Internally uses a JDNC driver which reads HTML.

Attributes

ClassName

Constructors

WebQueryEngine(String, String)

Attribute Detail

ClassName

```
protected static String ClassName = "fr.dyade.mediation.InfoExtractor.jdbc.IEDriver"
```

Constructor Detail

WebQueryEngine

```
public WebQueryEngine(String driverUrl, String queryStr)
```

Receive the URL name to query. We internally provide the Web JDBC Driver

Class *tcs.ain.infoserver.Worker*

```
public abstract class Worker
```

Extends:

tcs.ain.infoserver.InfoServerElement

Implements:

tcs.persist.TCSPersistent

Worker has its own thread and actually initiated actions within the InfoServer environment. Workers are typically defined by/for UserAgents and initiatiate interaction between other InfoServerElements. Base class handles default details of when the worker should do its work by using the Timer class. Timer is a special thread that is used to determine when the Worker should do its work. Subclasses must provide the doWork() method, which is called whenever the thread executes.

Attributes

- isRunning
- timer

Constructors

- Worker()

Operations

- doWork()
- getIsRunning()
- getTimer()
- setIsRunning(boolean)
- setTimer(Timer)
- start()
- stop()

Attribute Detail

isRunning

```
protected boolean isRunning = false
```

Internal flag for whether we are running; used when loading serialized image to auto-start

timer

```
protected Timer timer = null
```

The timer that drives when the worker does work.

Constructor Detail

Worker

```
public Worker()
```

Operation Detail

doWork

```
public void doWork()
```

This is the method called by my timer whenever I should do some work. Subclasses must define this.

getIsRunning

```
public boolean getIsRunning()
```

tgGet

getTimer

```
public Timer getTimer()
```

tgGet

setIsRunning

```
public void setIsRunning(boolean b)
```

tgSet

setTimer

```
public void setTimer(Timer timer)
```

tgSet

start

```
public void start()
```

stop

```
public void stop()
```

Interface Detail

Interface *tcs.ain.infoserver.access.IRemoteRepository*

```
public interface IRemoteRepository
```

Extends:

java.rmi.Remote

This interface provides access to user agents within the Info Server. May be replaced by a Jini Service in the near future.

Operations

addUserAgent(UserAgent)
getQueryEngine(String)
getQueryEngines()
getUserAgent(String)
getWorkerIdsFromAgent(String)
isConnected()
launchWorker(String, String, Hashtable)
launchWorker(String, String)

Operation Detail

addUserAgent

```
public abstract void addUserAgent(UserAgent ua)
```

Throws:
InfoServerException
RemoteException

getQueryEngine

```
public abstract QueryEngine getQueryEngine(String id)
```

Throws:
ElementNotFoundException
RemoteException

getQueryEngines

```
public abstract Hashtable getQueryEngines()
```

Throws:
RemoteException

getUserAgent

```
public abstract UserAgent getUserAgent(String id)
```

Throws:
ElementNotFoundException
RemoteException

getWorkerIdsFromAgent

```
public abstract Vector getWorkerIdsFromAgent(String id)
```

Throws:
RemoteException

isConnected

```
public abstract boolean isConnected()
```

Throws:
ElementNotFoundException
RemoteException

launchWorker

```
public abstract String launchWorker(String id, String workerId, Hashtable htParams)
```

Throws:
InfoServerException
RemoteException

launchWorker

```
public abstract String launchWorker(String id, String workerId)
```

Throws:
InfoServerException
RemoteException

Interface *tcs.ain.infoserver.IDataEventListener*

```
public interface IDataEventListener
```

Extends:
java.util.EventListener

Operations

cleanUp()

processData(DataEvent)

Operation Detail

cleanUp

```
public abstract void cleanUp()
```

processData

```
public abstract void processData(DataEvent evt)
```

Throws:
InfoServerException

Interface *tcs.ain.infoserver.IDataEventSource*

```
public interface IDataEventSource
```

Operations

addDataEventListener(IDataEventListener)
getDataEventListener(String)
getDataEventListeners()
removeDataEventListener(IDataEventListener)

Operation Detail

addDataEventListener

```
public abstract void addDataEventListener(IDataEventListener l)
```

getDataEventListener

```
public abstract IDataEventListener getDataEventListener(String id)
```

getDataEventListeners

```
public abstract Enumeration getDataEventListeners()
```

removeDataEventListener

```
public abstract void removeDataEventListener(IDataEventListener l)
```

Interface *tcs.ain.infoserver.IDataSource*

```
public interface IDataSource
```

This interface defines what to expect from an information supplier. Ideally a data source will provide: · some data · a definition of the data (DTD) · some ways to display the data The last is optional. The DTD is a good idea, but is not yet enforced.

Operations

getDTD()
getXMLStream()
getXSLDocCount()
getXSLDocument(String)
getXSLName(int)

Operation Detail

getDTD

```
public abstract DTD getDTD()
```

return a DTD describing the XML that can be retreived using the XML Stream.

getXMLStream

```
public abstract InputStream getXMLStream()
```

Return an input stream by which an XML document can be read. We return an INputStream rather than just a DOM XML document because: - some data may be too large to process as a single unit - this way multiple XML parsing techniques (SAX, DOM, etc) can be used at the discretion of the client. The XML Stream does NOT included DTD info.

Throws:
InfoServerException

getXSLDocCount

```
public abstract int getXSLDocCount()
```

How many XSL documents are provided for presenting the data

getXSLDocument

```
public abstract org.w3c.dom.Document getXSLDocument(String name)
```

The actual XSL Document with this name

getXSLName

```
public abstract String getXSLName(int idx)
```

The descriptive name for XSL document with this index position (0-based)

Interface *tcs.ain.infoserver.pqa.PQAConst*

```
public interface PQAConst
```

The Interface is a contains common constants which apply to the QueryNet PQA Interface. modified by S. Levine on 9/9/99

Attributes

CANCEL
DEBUGMODE
INVALIDLOGIN_TAG
LAUNCH
LOGIN_PATH
MENU_PATH
NOWORKERS_PATH
NO_RESULTS_MSG
RESULTS_TAG
RESULT_PATH
SECTION
USERNAME
USERNAME_TAG
WORKERID_TAG
WORKERLIST
WORKERLIST_TAG

Attribute Detail

CANCEL

```
String CANCEL = "CANCEL"
```

Form Cancel Button.

DEBUGMODE

```
boolean DEBUGMODE = Boolean.valueOf(QNetPQAResource.getString("DebugMode")).booleanValue()
```

Specifies whether app is in debug mode

INVALIDLOGIN_TAG

```
String INVALIDLOGIN_TAG = "<!-- INVALIDLOGINMSG -->"
```

Form Tag that lets up put dynamic Invalid Login Message in static html

LAUNCH

```
String LAUNCH = "LAUNCH"
```

Form Launch Button.

LOGIN_PATH

```
String LOGIN_PATH = QNetPQAResource.getString("LoginPath")
```

Path to Login page

MENU_PATH

```
String MENU_PATH = QNetPQAResource.getString("MenuPath")
```

Path to Menu page

NOWORKERS_PATH

```
String NOWORKERS_PATH = QNetPQAResource.getString("NoWorkersPath")
```

Path to No Workers page

NO_RESULTS_MSG

```
String NO_RESULTS_MSG = QNetPQAResource.getString("NoResultsMsg")
```

No Results Returned from QueryNet Message

RESULTS_TAG

```
String RESULTS_TAG = "<!-- RESULTS -->"
```

Form Tag that lets up put dynamic result information in static html

RESULT_PATH

```
String RESULT_PATH = QNetPQAResource.getString("ResultPath")
```

Path to Results page

SECTION

```
String SECTION = "SECTION"
```

Form Section Id Hidden Field.

USERNAME

```
String USERNAME = "USERNAME"
```

Form Hidden Field containing a UserId

USERNAME_TAG

```
String USERNAME_TAG = "<!-- USERNAME -->"
```

Form Tag that lets up put dynamic UserName information in static html

WORKERID_TAG

```
String WORKERID_TAG = "<!-- WORKERID -->"
```

Form Tag that lets up put dynamic WorkerId information in static html

WORKERLIST

```
String WORKERLIST = "WORKERLIST"
```

Form Listbox containing a list of worker ids.

WORKERLIST_TAG

```
String WORKERLIST_TAG = "<!-- WORKERLIST -->"
```

Form Tag that lets up put dynamic WorkerList ListBox in static html

Interface *tcs.ain.infoserver.servlet.ISElementCreator*

```
public interface ISElementCreator
```

Attributes

CLASS
CUSTOMSOURCE_ET
DELIMIT
DESCRIPTION
DESTINATION_ET
FORMATTER
NAME
REQUIRED
STYLES

Operations

getCustomSource(String, HttpServletRequest, StringBuffer)
getDestination(String, Vector, StringBuffer)
getElementNames(int, boolean)
getFormatStyle(UserDeliveryDestination)
getFormatter(QueryEngine, String)
getRequiredData(String, InfoServerElement, UserAgent, int)
getRequirements(String, int)
getSubForm(String, InfoServerElement, UserAgent, int)
loadElements(String, String)

Attribute Detail

CLASS

```
String CLASS = "CLASS"
```

Element Attribute

CUSTOMSOURCE_ET

```
int CUSTOMSOURCE_ET = 0
```

Element Type

DELIMIT

```
String DELIMIT = "~"
```

String Delimiter

DESCRIPTION

```
String DESCRIPTION = "DESC"
```

Element Attribute

DESTINATION_ET

```
int DESTINATION_ET = 1
```

Element Type

FORMATTER

```
String FORMATTER = "FORMATTER"
```

Element Attribute

NAME

```
String NAME = "NAME"
```

Element Attribute

REQUIRED

```
String REQUIRED = "REQUIRED"
```

Element Attribute

STYLES

```
String STYLES = "STYLES"
```

Element Attribute

Operation Detail

getCustomSource

```
public abstract QueryEngine getCustomSource(String source, HttpServletRequest request, StringBuffer
errorMsg)
```

getDestination

```
public abstract UserDeliveryDestination getDestination(String destination, Vector reqInfo,
StringBuffer errorMsg)
```

getElementNames

```
public abstract Vector getElementNames(int elementType, boolean addDesc)
```

getFormatStyle

```
public abstract String getFormatStyle(UserDeliveryDestination dest)
```

getFormatter

```
public abstract DataFormatter getFormatter(QueryEngine ds, String style)
```

getRequiredData

```
public abstract Vector getRequiredData(String destId, InfoServerElement element, UserAgent agent,
int elementType)
```

getRequirements

```
public abstract Vector getRequirements(String name, int elementType)
```

getSubForm

```
public abstract String getSubForm(String name, InfoServerElement element, UserAgent agent, int
elementType)
```

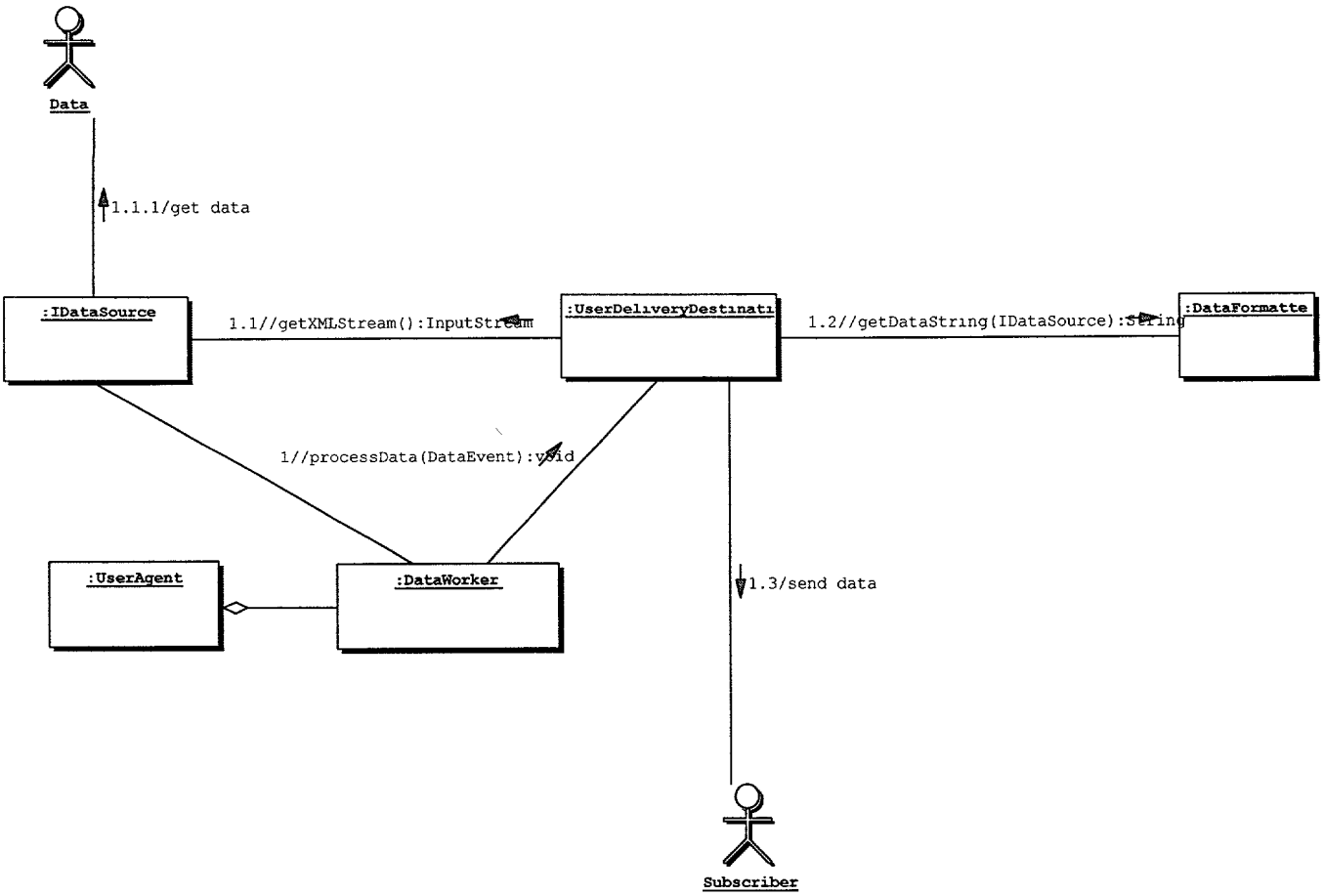
loadElements

```
public abstract void loadElements(String sourceElements, String destElements)
```

Load elements descriptions into hashtables

Interaction Diagrams

Collaboration Diagram *overview*



basic component relationships

Object Summary

- Data
- Subscriber

Object Detail

Object

Instantiates:
tcs.ain.infoserver.IDataSource

Message Detail

get data to **Object** Data

Number:
1.1.1

Synchronization:
call

processingDuration:
50

sendingInstant:
175

Object

Instantiates:
tcs.ain.infoserver.UserAgent

"Aggregates" links

to **Object**

Object

Instantiates:
tcs.ain.infoserver.DataFormatter

Object

Instantiates:
tcs.ain.infoserver.UserDeliveryDestination

Message Detail

to **Object**

Number:
1.1

Synchronization:
call

processingDuration:
80

sendingInstant:
155

Operation:
tcs.ain.infoserver.IDataSource.getXMLStream()

operationNameAsText:
'getXMLStream():InputStream'

send data to **Object** Subscriber

Number:
1.3

Synchronization:
call

processingDuration:
50

sendingInstant:
325

to **Object**

Number:
1.2

Synchronization:
call

processingDuration:
50

sendingInstant:
255

Operation:
tcs.ain.infoserver.DataFormatter.getDataString(tcs.ain.infoserver.IDataSource)

operationNameAsText:
'getDataString(IDataSource):String'

Object

Instantiates:
tcs.ain.infoserver.DataWorker

multiple instances

Message Detail

to **Object**

Number:
1

Synchronization:
call

processingDuration:
250

sendingInstant:
135

Operation:
tcs.ain.infoserver.UserDeliveryDestination.processData(tcs.ain.infoserver.DataEvent)

operationNameAsText:
'processData(DataEvent):void'

"Associates" links

to **Object**

Object *Data*

Stereotype:
actor

Object *Subscriber*

Stereotype:
actor

Package *tcs.ain.infoserver.access*

Class Diagrams

diagram access

Classes

class tcs.ain.infoserver.access.InfoServerAccess
class tcs.ain.infoserver.access.InfoServerAccess_Skel
class tcs.ain.infoserver.access.InfoServerAccess_Stub

Interfaces

interface tcs.ain.infoserver.access.IRemoteRepository

Class Diagrams

Class Diagram *access*



Class Nodes

tcs.ain.infoserver.access.InfoServerAccess
tcs.ain.infoserver.access.InfoServerAccess_Skel
tcs.ain.infoserver.access.InfoServerAccess_Stub

Interface Nodes

tcs.ain.infoserver.access.IRemoteRepository

Class Node Detail

Class *tcs.ain.infoserver.access.InfoServerAccess*

This class provides access to user agents within the Info Server. May be replaced by a Jini Service in the near future. This provides access to the assigned Repository

Extends:
UnicastRemoteObject

Implements:
IRemoteRepository

Implementation links

to **Interface** IRemoteRepository

Class *tcs.ain.infoserver.access.InfoServerAccess_Skel*

Extends:
java.lang.Object

Implements:
java.rmi.server.Skeleton

Class *tcs.ain.infoserver.access.InfoServerAccess_Stub*

Extends:
java.rmi.server.RemoteStub

Implements:
tcs.ain.infoserver.access.IRemoteRepository
java.rmi.Remote

Implementation links

to **Interface** IRemoteRepository

Interface Node Detail

Interface *tcs.ain.infoserver.access.IRemoteRepository*

This interface provides access to user agents within the Info Server. May be replaced by a Jini Service in the near future.

Extends:
java.rmi.Remote

Class Detail

Class *tcs.ain.infoserver.access.InfoServerAccess*

```
public class InfoServerAccess
```

Extends:

java.rmi.server.UnicastRemoteObject

Implements:

tcs.ain.infoserver.access.IRemoteRepository

This class provides access to user agents within the Info Server. May be replaced by a Jini Service in the near future. This provides access to the assigned Repository

Attributes

repository

Constructors

InfoServerAccess(Repository)

Operations

addUserAgent(UserAgent)
getQueryEngine(String)
getQueryEngines()
getUserAgent(String)
getWorkerIdsFromAgent(String)
isConnected()
launchWorker(String, String, Hashtable)
launchWorker(String, String)

Attribute Detail

repository

```
protected Repository repository
```

The repository which we represent.

Constructor Detail

InfoServerAccess

```
public InfoServerAccess(Repository r)
```

Construct with this repository and register the 'InfoServerAccess' name.

Throws:

RemoteException
MalformedURLException

Operation Detail

addUserAgent

```
public void addUserAgent(UserAgent ua)
```

Pass through the add, but also force a save of all agents.

Throws:

InfoServerException
RemoteException

getQueryEngine

```
public QueryEngine getQueryEngine(String id)
```

Direct passthrough for the get

Throws:

RemoteException

getQueryEngines

```
public Hashtable getQueryEngines()
```

Direct passthrough for the get

Throws:
RemoteException

getUserAgent

```
public UserAgent getUserAgent(String id)
```

Direct passthrough for the get

Throws:
RemoteException

getWorkerIdsFromAgent

```
public Vector getWorkerIdsFromAgent(String agentId)
```

Return all id's associated with a specified agent

Throws:
RemoteException

isConnected

```
public boolean isConnected()
```

Used for testing an RMI Connection. Always sends true. If client calls this and no connection has been establishished then they will get an exception

Throws:
ElementNotFoundException
RemoteException

launchWorker

```
public String launchWorker(String agentId, String workerId, Hashtable htParams)
```

Call the specidied workers doWork method, pass in parameters.

Throws:
InfoServerException
RemoteException

launchWorker

```
public String launchWorker(String agentId, String workerId)
```

Call the specidied workers doWork method

Throws:
InfoServerException
RemoteException

Class *tcs.ain.infoserver.access.InfoServerAccess_Skel*

```
public final class InfoServerAccess_Skel
```

Extends:
java.lang.Object

Implements:
java.rmi.server.Skeleton

Attributes

interfaceHash
operations

Constructors

InfoServerAccess_Skel()

Operations

dispatch(java.rmi.Remote, java.rmi.server.RemoteCall, int, long)
getOperations()

Attribute Detail

interfaceHash

```
private static final long interfaceHash
```

operations

```
private static final java.rmi.server.Operation[] operations
```

Constructor Detail

InfoServerAccess_Skel

```
public InfoServerAccess_Skel()
```

Operation Detail

dispatch

```
public void dispatch(java.rmi.Remote, java.rmi.server.RemoteCall, int, long)
```

getOperations

```
public java.rmi.server.Operation[] getOperations()
```

Class *tcs.ain.infoserver.access.InfoServerAccess_Stub*

```
public final class InfoServerAccess_Stub
```

Extends:

java.rmi.server.RemoteStub

Implements:

tcs.ain.infoserver.access.IRemoteRepository
java.rmi.Remote

Attributes

interfaceHash
operations

Constructors

InfoServerAccess_Stub()
InfoServerAccess_Stub(java.rmi.server.RemoteRef)

Operations

addUserAgent(tcs.ain.infoserver.UserAgent)
getQueryEngine(java.lang.String)
getQueryEngines()
getUserAgent(java.lang.String)
getWorkerIdsFromAgent(java.lang.String)
isConnected()
launchWorker(java.lang.String, java lang.String, java.util.Hashtable)
launchWorker(java.lang.String, java.lang.String)

Attribute Detail

interfaceHash

```
private static final long interfaceHash
```

operations

```
private static final java.rmi.server.Operation[] operations
```

Constructor Detail

InfoServerAccess_Stub

```
public InfoServerAccess_Stub()
```

InfoServerAccess_Stub

```
public InfoServerAccess_Stub(java.rmi.server.RemoteRef)
```

Operation Detail

addUserAgent

```
public void addUserAgent(tcs.ain.infoserver.UserAgent)
```

getQueryEngine

```
public tcs.ain.infoserver.QueryEngine getQueryEngine(java.lang.String)
```

getQueryEngines

```
public java.util.Hashtable getQueryEngines()
```

getUserAgent

```
public tcs.ain.infoserver.UserAgent getUserAgent(java.lang.String)
```

getWorkerIdsFromAgent

```
public java.util.Vector getWorkerIdsFromAgent(java.lang.String)
```

isConnected

```
public boolean isConnected()
```

launchWorker

```
public java.lang.String launchWorker(java.lang.String, java.lang.String, java.util.Hashtable)
```

launchWorker

```
public java.lang.String launchWorker(java.lang.String, java.lang.String)
```

Interface Detail

Interface *tcs.ain.infoserver.access.IRemoteRepository*

```
public interface IRemoteRepository
```

Extends:

java.rmi.Remote

This interface provides access to user agents within the Info Server. May be replaced by a Jini Service in the near future.

Operations

addUserAgent(UserAgent)
getQueryEngine(String)
getQueryEngines()
getUserAgent(String)
getWorkerIdsFromAgent(String)
isConnected()

launchWorker(String, String, Hashtable)
launchWorker(String, String)

Operation Detail

addUserAgent

```
public abstract void addUserAgent(UserAgent ua)
```

Throws:
InfoServerException
RemoteException

getQueryEngine

```
public abstract QueryEngine getQueryEngine(String id)
```

Throws:
ElementNotFoundException
RemoteException

getQueryEngines

```
public abstract Hashtable getQueryEngines()
```

Throws:
RemoteException

getUserAgent

```
public abstract UserAgent getUserAgent(String id)
```

Throws:
ElementNotFoundException
RemoteException

getWorkerIdsFromAgent

```
public abstract Vector getWorkerIdsFromAgent(String id)
```

Throws:
RemoteException

isConnected

```
public abstract boolean isConnected()
```

Throws:
ElementNotFoundException
RemoteException

launchWorker

```
public abstract String launchWorker(String id, String workerId, Hashtable htParams)
```

Throws:
InfoServerException
RemoteException

launchWorker

```
public abstract String launchWorker(String id, String workerId)
```

Throws:
InfoServerException
RemoteException

Package *tcs.ain.infoserver.dest*

Class Diagrams

diagram dest

Class Diagrams

Class Diagram *dest*

Package *tcs.ain.infoserver.pqa*

Class Diagrams

diagram pqa

Classes

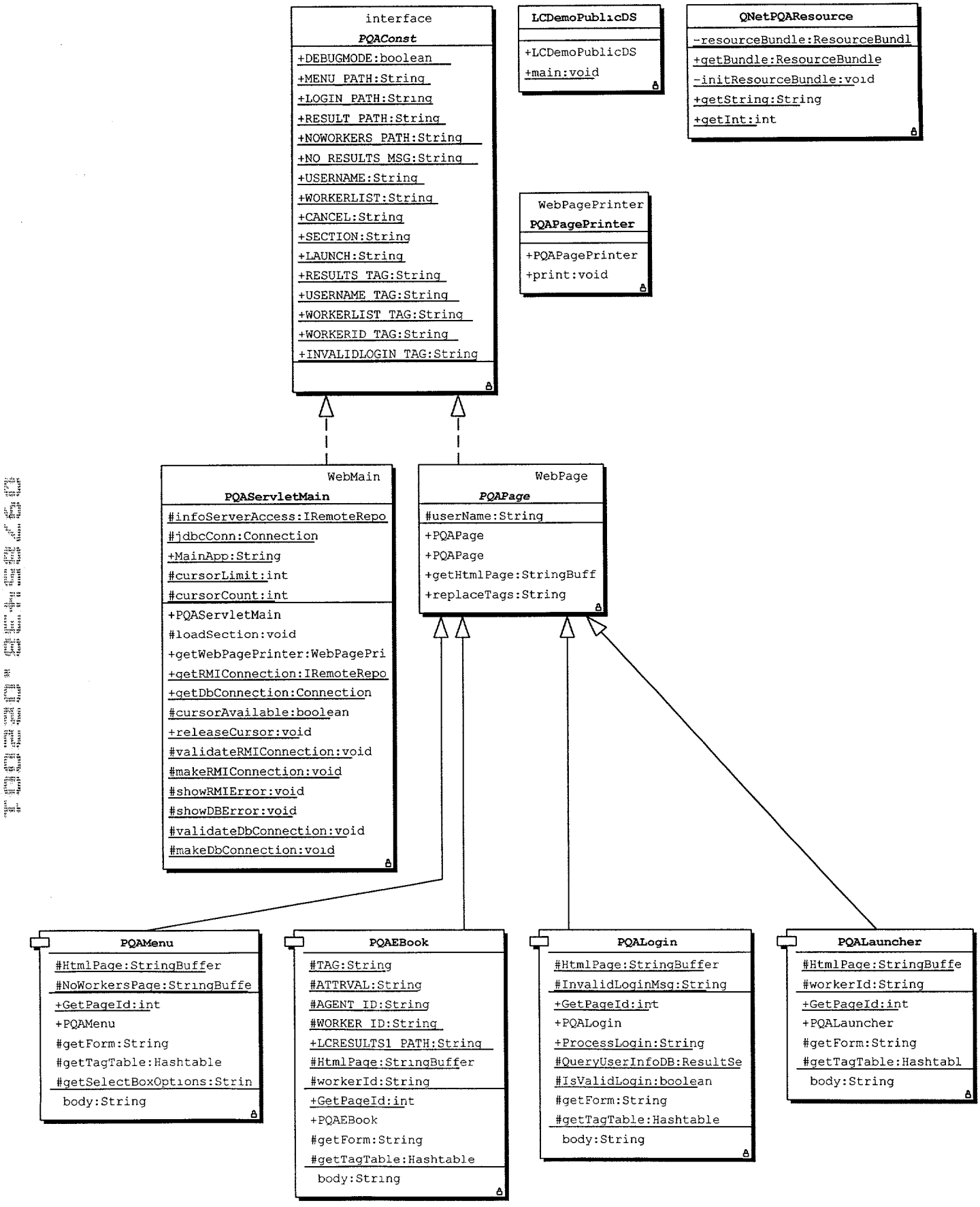
class tcs.ain.infoserver.pqa.LCDemoPublicDS
class tcs.ain.infoserver.pqa.PQAEBook
class tcs.ain.infoserver.pqa.PQALauncher
class tcs.ain.infoserver.pqa.PQALogin
class tcs.ain.infoserver.pqa.PQAMenu
class tcs.ain.infoserver.pqa.PQAPage
class tcs.ain.infoserver.pqa.PQAPagePrinter
class tcs.ain.infoserver.pqa.PQAServletMain
class tcs.ain.infoserver.pqa.QNetPQAResource

Interfaces

interface tcs.ain.infoserver.pqa.PQAConst

Class Diagrams

Class Diagram *pqa*


interface
PQAConst
+DEBUGMODE:boolean
+MENU_PATH:String
+LOGIN_PATH:String
+RESULT_PATH:String
+NOWORKERS_PATH:String
+NO_RESULTS_MSG:String
+USERNAME:String
+WORKERLIST:String
+CANCEL:String
+SECTION:String
+LAUNCH:String
+RESULTS_TAG:String
+USERNAME_TAG:String
+WORKERLIST_TAG:String
+WORKERID_TAG:String
+INVALIDLOGIN_TAG:String
LCDemoPublicDS
+LCDemoPublicDS
+main:void
QNetPQAResource
-resourceBundle:ResourceBundl
+getBundle:ResourceBundle
-initResourceBundle:void
+getString:String
+getInt:int
WebPagePrinter
PQAPagePrinter
+PQAPagePrinter
+print:void
WebMain
PQAServletMain
#infoServerAccess:IRemoteRepo
#jdbcConn:Connection
+MainApp:String
#cursorLimit:int
#cursorCount:int
+PQAServletMain
#loadSection:void
+getWebPagePrinter:WebPagePri
+getRMIConnection:IRemoteRepo
+getDbConnection:Connection
#cursorAvailable:boolean
+releaseCursor:void
#validateRMIConnection:void
#makeRMIConnection:void
#showRMIError:void
#showDBError:void
#validateDbConnection:void
#makeDbConnection:void
WebPage
PQAPage
#userName:String
+PQAPage
+PQAPage
+getHtmlPage:StringBuff
+replaceTags:String
PQAMenu
#HtmlPage:StringBuffer
#NoWorkersPage:StringBuffe
+GetPageId:int
+PQAMenu
#getForm:String
#getTagTable:Hashtable
#getSelectBoxOptions:Strin
body:String
PQAEBook
#TAG:String
#ATTRVAL:String
#AGENT_ID:String
#WORKER_ID:String
+LCRESULTS1_PATH:String
#HtmlPage:StringBuffer
#workerId:String
+GetPageId:int
+PQAEBook
#getForm:String
#getTagTable:Hashtable
body:String
PQALogin
#HtmlPage:StringBuffer
#InvalidLoginMsg:String
+GetPageId:int
+PQALogin
+ProcessLogin:String
#QueryUserInfoDB:ResultSe
#IsValidLogin:boolean
#getForm:String
#getTagTable:Hashtable
body:String
PQALauncher
#HtmlPage:StringBuffe
#workerId:String
+GetPageId:int
+PQALauncher
#getForm:String
#getTagTable:Hashtabl
body:String

Class Nodes

tcs.ain.infoserver.pqa.LCDemoPublicDS
tcs.ain.infoserver.pqa.PQAEBook
tcs.ain.infoserver.pqa.PQALauncher
tcs.ain.infoserver.pqa.PQALogin
tcs.ain.infoserver.pqa.PQAMenu
tcs.ain.infoserver.pqa.PQAPage
tcs.ain.infoserver.pqa.PQAPagePrinter
tcs.ain.infoserver.pqa.PQAServletMain
tcs.ain.infoserver.pqa.QNetPQAResource

Interface Nodes

tcs.ain.infoserver.pqa.PQAConst

Class Node Detail

Class *tcs.ain.infoserver.pqa.LCDemoPublicDS*

Tests the IMAP Query Engine; query every one minute

Class *tcs.ain.infoserver.pqa.PQAEBook*

Launch a DataWorker for a given Agent, and return results

Extends:
PQAPage

Generalization links

to **Class** PQAPage

Class *tcs.ain.infoserver.pqa.PQALauncher*

Launch a DataWorker for a given Agent, and return results

Extends:
PQAPage

Generalization links

to **Class** PQAPage

Class *tcs.ain.infoserver.pqa.PQALogin*

USER LOGIN PAGE. This prompts user for username/pw and tries and to find (or create) a UserAgent object for them. The UserAgent object is inserted into the user's Session under the name 'userAgent', for retention between screens This will eventually validate against and reference the SMS Internet Suite UserInfo database. Note: New agents are created for authenticated users, and their WebProv values will be assigned to the agent for name, email, min We will need to keep the agent info up-to-date somehow.

Extends:
PQAPage

Generalization links

to **Class** PQAPage

Class *tcs.ain.infoserver.pqa.PQAMenu*

Return a Menu listing of all DataWorkers belonging to a given UserAgent.

Extends:
PQAPage

Generalization links

to **Class** PQAPage

Class *tcs.ain.infoserver.pqa.PQAPage*

Launch a DataWorker for a given Agent, and return results

Extends:
WebPage

Implements:
PQAConst

Implementation links

to **Interface** PQAConst

Class *tcs.ain.infoserver.pqa.PQAPagePrinter*

Base class for web page printing operations. Provides methods to print the header, banner, etc. Subclasses can provide distinct color / font / banner schemes for individual projects or pages

Extends:
WebPagePrinter

Class *tcs.ain.infoserver.pqa.PQAServletMain*

This is the Main page for the QueryNet PQA Servlet. This loads pages according to the received Section variable

Extends:
WebMain

Implements:
PQAConst

Implementation links

to **Interface** PQAConst

Class *tcs.ain.infoserver.pqa.QNetPQAResource*

This class is a singleton for a property resource bundle to be used by the QueryNet PQA Servlet. The resource bundle contains values that could potentially be displayed to users. It also provides a direct, static method for safely getting String values without worrying about MIssingResourceExceptions ras; 2/29/99; modified by S. Levine on 5/01/00

Interface Node Detail

Interface *tcs.ain.infoserver.pqa.PQAConst*

The Interface is a contains common constants which apply to the QueryNet PQA Interface. modified by S. Levine on 9/9/99

Class Detail

Class *tcs.ain.infoserver.pqa.LCDemoPublicDS*

```
public class LCDemoPublicDS
```

Tests the IMAP Query Engine; query every one minute

Constructors

LCDemoPublicDS()

Operations

main(String[])

Constructor Detail

LCDemoPublicDS

```
public LCDemoPublicDS()
```

Throws:

Exception

Operation Detail

main

```
public static void main(String[] args)
```

Class *tcs.ain.infoserver.pqa.PQAEBook*

```
public class PQAEBook
```

Extends:

tcs.ain.infoserver.pqa.PQAPage

Launch a DataWorker for a given Agent, and return results

Attributes

AGENT_ID
ATTRVAL
HtmlPage
LCRESULTS1_PATH
TAG
workerId
WORKER_ID

Constructors

PQAEBook(HttpServletRequest, HttpServletResponse, String)

Operations

getBody()
getForm(String)
GetPageId()
getTagTable(String)

Attribute Detail

AGENT_ID

```
protected static final String AGENT_ID = "rsmith"
```

Agent ID

ATTRVAL

```
protected static final String ATTRVAL = "SF_ATTRSTR"
```

Attribute Value parameter name constant

HtmlPage

```
protected static StringBuffer HtmlPage = null
```

Stores Html Page associated with a form producing class

LCRESULTS1_PATH

```
public static final String LCRESULTS1_PATH = QNetPQAResource.getString("LCResults1")
```

Results Page Path

TAG

```
protected static final String TAG = "SF_TAG"
```

Tag parameter name constant

workerId

```
protected String workerId = ""
```

Stores Html Page associated with a form producing class

WORKER_ID

```
protected static final String WORKER_ID = "LC Operation Manual"
```

WorkerID

Constructor Detail

PQAEBook

```
public PQAEBook(HttpServletRequest request, HttpServletResponse response, String userName)
```

Constructor calls super() and sets the title for this page

Operation Detail

getBody

```
public String getBody()
```

get Login Body. Since the initial login form is stored on the pqa, this should only be called to retrieve an error login

getForm

```
protected String getForm(String results)
```

Create an html form for selecting or removing existing Data Workers. This is the initial form for the Query Net Wizard

GetPageId

```
public static int GetPageId()
```

getTagTable

```
protected Hashtable getTagTable(String results)
```

Create a table of tags and their values to be used in replacing tags found in static html pages. Used to add dynamic content to these pages.

Class *tcs.ain.infoserver.pqa.PQALauncher*

```
public class PQALauncher
```

Extends:

tcs.ain.infoserver.pqa.PQAPage

Launch a DataWorker for a given Agent, and return results

Attributes

HtmlPage
workerId

Constructors

PQALauncher(HttpServletRequest, HttpServletResponse, String)

Operations

getBody()
getForm(String)
GetPageId()
getTagTable(String)

Attribute Detail

HtmlPage

```
protected static StringBuffer HtmlPage = null
```

Stores Html Page associated with a form producing class

workerId

```
protected String workerId = ""
```

Stores Html Page associated with a form producing class

Constructor Detail

PQALauncher

```
public PQALauncher(HttpServletRequest request, HttpServletResponse response, String userName)
```

Constructor calls super() and sets the title for this page

Operation Detail

getBody

```
public String getBody()
```

get Login Body. Since the initial login form is stored on the pqa, this should only be called to retrieve an error login

getForm

```
protected String getForm(String results)
```

Create an html form for selecting or removing existing Data Workers. This is the initial form for the Query Net Wizard

GetPageId

```
public static int GetPageId()
```

getTagTable

```
protected Hashtable getTagTable(String results)
```

Create a table of tags and their values to be used in replacing tags found in static html pages. Used to add dynamic content to these pages.

Class *tcs.ain.infoserver.pqa.PQALogin*

```
public class PQALogin
```

Extends:

tcs.ain.infoserver.pqa.PQAPage

USER LOGIN PAGE. This prompts user for username/pw and tries and to find (or create) a UserAgent object for them. The UserAgent object is inserted into the user's Session under the name 'userAgent', for retention between screens This will eventually validate against and reference the SMS Internet Suite UserInfo database. Note: New agents are created for authenticated users, and their WebProv values will be assigned to the agent for name, email, min We will need to keep the agent info up-to-date somehow.

Attributes

HtmlPage
InvalidLoginMsg

Constructors

PQALogin(HttpServletRequest, HttpServletResponse)

Operations

getBody()
getForm()
GetPageId()
getTagTable()
IsValidLogin(ResultSet, String)
ProcessLogin(HttpServletRequest)
QueryUserInfoDB(String)

Attribute Detail

HtmlPage

```
protected static StringBuffer HtmlPage = null
```

Stores Html Page associated with a form producing class

InvalidLoginMsg

```
protected static final String InvalidLoginMsg = QNetPQAResource.getString("InvalidLoginMsg")
```

Stores Html Page associated with a form producing class

Constructor Detail

PQALogin

```
public PQALogin(HttpServletRequest request, HttpServletResponse response)
```

Constructor calls super() and sets the title for this page

Operation Detail

getBody

```
public String getBody()
```

get Login Body. Since the initial login form is stored on the pqa, this should only be called to retrieve an error login

getForm

```
protected String getForm()
```

Create an html form for selecting or removing existing Data Workers. This is the initial form for the Query Net Wizard

GetPageId

```
public static int GetPageId()
```

getTagTable

```
protected Hashtable getTagTable()
```

Create a table of tags and their values to be used in replacing tags found in static html pages. Used to add dynamic content to these pages.

IsValidLogin

```
protected static boolean IsValidLogin(ResultSet r, String password)
```

Autherticate the user.

ProcessLogin

```
public static String ProcessLogin(HttpServletRequest req)
```

Validate a login, return username if valid log in, null if invalid. REMEMBER: users can log in with Alias or PIN so be sure

QueryUserInfoDB

```
protected static ResultSet QueryUserInfoDB(String username)
```

Query the UserProvisioning database for the login name

Class *tcs.ain.infoserver.pqa.PQAMenu*

```
public class PQAMenu
```

Extends:

tcs.ain.infoserver.pqa.PQAPage

Return a Menu listing of all DataWorkers belonging to a given UserAgent.

Attributes

HtmlPage
NoWorkersPage

Constructors

PQAMenu(HttpServletRequest, HttpServletResponse, String)

Operations

getBody()
getForm(Vector)
GetPageId()
getSelectBoxOptions(Vector)
getTagTable(Vector)

Attribute Detail

HtmlPage

```
protected static StringBuffer HtmlPage = null
```

Stores Html Page associated with a form producing class

NoWorkersPage

```
protected static StringBuffer NoWorkersPage = null
```

Stores Html Page associated with a form producing class

Constructor Detail

PQAMenu

```
public PQAMenu(HttpServletRequest request, HttpServletResponse response, String userName)
```

Constructor calls super()

Operation Detail

getBody

```
public String getBody()
```

get Menu Body. List all DataWorkers associated with a given agent

getForm

```
protected String getForm(Vector workerList)
```

Create an html form for selecting or removing existing Data Workers. This is the initial form for the Query Net Wizard

GetPageId

```
public static int GetPageId()
```

getSelectBoxOptions

```
protected String getSelectBoxOptions(Vector workerList)
```

Create a select box for the given parameter and select a specified value. *

getTagTable

```
protected Hashtable getTagTable(Vector workerList)
```

Create a table of tags and their values to be used in replacing tags found in static html pages. Used to add dynamic content to these pages.

Class *tcs.ain.infoserver.pqa.PQAPage*

```
public abstract class PQAPage
```

Extends:

tcs.web.WebPage

Implements:

tcs.ain.infoserver.pqa.PQAConst

Launch a DataWorker for a given Agent, and return results

Attributes

userName

Constructors

PQAPage(HttpServletRequest, HttpServletResponse)
PQAPage(HttpServletRequest, HttpServletResponse, String)

Operations

getHtmlPage(String)
replaceTags(String, Hashtable)

Attribute Detail

userName

```
protected String userName
```

Constructor Detail

PQAPage

```
public PQAPage(HttpServletRequest request, HttpServletResponse response)
```

Constructor calls super() and sets the title for this page

PQAPage

```
public PQAPage(HttpServletRequest request, HttpServletResponse response, String userName)
```

Constructor calls super() and sets the title for this page

Operation Detail

getHtmlPage

```
public StringBuffer getHtmlPage(String path)
```

Read in specified Html Page and store to static HtmlPage StringBuffer

Throws:
Exception

replaceTags

```
public String replaceTags(String content, Hashtable ht)
```

Read in specified Html Page and store content to StringBuffer

Class *tcs.ain.infoserver.pqa.PQAPagePrinter*

```
public class PQAPagePrinter
```

Extends:
tcs.web.WebPagePrinter

Base class for web page printing operations. Provides methods to print the header, banner, etc. Subclasses can provide distinct color / font / banner schemes for individual projects or pages

Constructors

PQAPagePrinter(WebPage)

Operations

print()

Constructor Detail

PQAPagePrinter

```
public PQAPagePrinter(WebPage page)
```

Construct the page printer object. This assings the owner and gets the output stream

Operation Detail

print

```
public void print()
```

public print method; what client calls to start the printing process. This calls the basic components of the page (header, banner, body, footer)

Class *tcs.ain.infoserver.pqa.PQAServletMain*

```
public class PQAServletMain
```

Extends:
tcs.web.WebMain

Implements:
tcs.ain.infoserver.pqa.PQAConst

This is the Main page for the QueryNet PQA Servlet. This loads pages according to the received Section variable

Attributes

cursorCount
cursorLimit
infoServerAccess
jdbcConn
MainApp

Constructors

PQAServletMain()

Operations

cursorAvailable()
getDbConnection()

getRMIConnection()
getWebPagePrinter(WebPage)
loadSection(HttpServletRequest, HttpServletResponse)
makeDbConnection()
makeRMIConnection()
releaseCursor()
showDBError(HttpServletResponse)
showRMIError(HttpServletResponse)
validateDbConnection()
validateRMIConnection()

Attribute Detail

cursorCount

```
protected static int cursorCount = 0
```

The database cursors currently open by repository.

cursorLimit

```
protected static final int cursorLimit = QNetPQAResource.getInt("DbCursorLimit")
```

The database cursor limit for the Repository database connection.

infoServerAccess

```
protected static IRemoteRepository infoServerAccess = null
```

Remote InfoServer Access Repository for loading / saving User Agents and Sources. Accessed via RMI

jdbcConn

```
protected static Connection jdbcConn
```

JDBC Connection for querying provisioning information from database

MainApp

```
public static String MainApp = QNetPQAResource.getString("MainAppPath")
```

Constructor Detail

PQAServletMain

```
public PQAServletMain()
```

Operation Detail

cursorAvailable

```
protected static boolean cursorAvailable()
```

Check to see if cursor limit has been reached

getDbConnection

```
public static Connection getDbConnection()
```

Return a reference to jdbcConn if the CursorLimit has not been exceeded and a valid connection can be established, otherwise return null. This protects from exceeding the a given Database Implementation's cursor limit setting. It is the calling procedures responsibility to call release cursor when finished using the jdbcConn reference. Throws exception if unable to establish a valid database connection

Throws:
SQLException

getRMIConnection

```
public static IRemoteRepository getRMIConnection()
```

Connect the remote object to the server if necessary and return RMI Connection

Throws:
Exception

getWebPagePrinter

```
public WebPagePrinter getWebPagePrinter(WebPage wp)
```

Connect the remote object to the server if necessary and return RMI Connection

Throws:
Exception

loadSection

```
protected void loadSection(HttpServletRequest request, HttpServletResponse response)
```

Evaluate the 'Section' value and load the appropriate page. If Section is not set, then just load the MONTHVIEW_PAGE page

makeDbConnection

```
protected static void makeDbConnection()
```

Make sure jdbcConn is valid. If not, report the error to out and return false

Throws:
SQLException

makeRMIConnection

```
protected static void makeRMIConnection()
```

Connect the remote object to the server if necessary

Throws:
Exception

releaseCursor

```
public static void releaseCursor()
```

Decrement cursorCount so that another process may reference jdbcConn

showDBError

```
protected static void showDBError(HttpServletResponse response)
```

Display a spiffy RMI error message

showRMIError

```
protected static void showRMIError(HttpServletResponse response)
```

Display a spiffy RMI error message

validateDbConnection

```
protected static void validateDbConnection()
```

Make sure con is valid. If not, report the error to out and throw the exception

Throws:
SQLException

validateRMIConnection

```
protected static void validateRMIConnection()
```

make connection if necessary; test the connection; if failed, try to to remake the connection return true if good connection

Class *tcs.ain.infoserver.pqa.QNetPQAResource*

```
public class QNetPQAResource
```

This class is a singleton for a property resource bundle to be used by the QueryNet PQA Servlet. The resource bundle contains values that could potentially be displayed to users. It also provides a direct, static method for safely getting String values without worrying about MIssingResourceExceptions ras; 2/29/99; modified by S. Levine on 5/01/00

Attributes

resourceBundle

Operations

getBundle()
getInt(String)
getString(String)
initResourceBundle()

Attribute Detail

resourceBundle

```
private static ResourceBundle resourceBundle = null
```

The resource bundle we keep for use by the package.

Operation Detail

getBundle

```
public static ResourceBundle getBundle()
```

Return the single, static instance of the resource bundle for the SMPP Module

getInt

```
public static int getInt(String key)
```

Pass through to the resourceBundle, except that we internally catch the MissingResourceException and report the error but return a "?". Return Int value.

getString

```
public static String getString(String key)
```

Pass through to the resourceBundle, except that we internally catch the MissingResourceException and report the error but return a "?"

initResourceBundle

```
private static void initResourceBundle()
```

Create the resource bundle

Interface Detail

Interface *tcs.ain.infoserver.pqa.PQAConst*

```
public interface PQAConst
```

The Interface is a contains common constants which apply to the QueryNet PQA Interface. modified by S. Levine on 9/9/99

Attributes

CANCEL
DEBUGMODE
INVALIDLOGIN_TAG
LAUNCH
LOGIN_PATH
MENU_PATH
NOWORKERS_PATH
NO_RESULTS_MSG
RESULTS_TAG
RESULT_PATH
SECTION

USERNAME
USERNAME_TAG
WORKERID_TAG
WORKERLIST
WORKERLIST_TAG

Attribute Detail

CANCEL

```
String CANCEL = "CANCEL"
```

Form Cancel Button.

DEBUGMODE

```
boolean DEBUGMODE = Boolean.valueOf(QNetPQAResource.getString("DebugMode")).booleanValue()
```

Specifies whether app is in debug mode

INVALIDLOGIN_TAG

```
String INVALIDLOGIN_TAG = "<!-- INVALIDLOGINMSG -->"
```

Form Tag that lets up put dynamic Invalid Login Message in static html

LAUNCH

```
String LAUNCH = "LAUNCH"
```

Form Launch Button.

LOGIN_PATH

```
String LOGIN_PATH = QNetPQAResource.getString("LoginPath")
```

Path to Login page

MENU_PATH

```
String MENU_PATH = QNetPQAResource.getString("MenuPath")
```

Path to Menu page

NOWORKERS_PATH

```
String NOWORKERS_PATH = QNetPQAResource.getString("NoWorkersPath")
```

Path to No Workers page

NO_RESULTS_MSG

```
String NO_RESULTS_MSG = QNetPQAResource.getString("NoResultsMsg")
```

No Results Returned from QueryNet Message

RESULTS_TAG

```
String RESULTS_TAG = "<!-- RESULTS -->"
```

Form Tag that lets up put dynamic result information in static html

RESULT_PATH

```
String RESULT_PATH = QNetPQAResource.getString("ResultPath")
```

Path to Results page

SECTION

```
String SECTION = "SECTION"
```

Form Section Id Hidden Field.

USERNAME

```
String USERNAME = "USERNAME"
```

Form Hidden Field containing a UserId

USERNAME_TAG

```
String USERNAME_TAG = "<!-- USERNAME -->"
```

Form Tag that lets up put dynamic UserName information in static html

WORKERID_TAG

```
String WORKERID_TAG = "<!-- WORKERID -->"
```

Form Tag that lets up put dynamic WorkerId information in static html

WORKERLIST

```
String WORKERLIST = "WORKERLIST"
```

Form Listbox containing a list of worker ids.

WORKERLIST_TAG

```
String WORKERLIST_TAG = "<!-- WORKERLIST -->"
```

Form Tag that lets up put dynamic WorkerList ListBox in static html

Package *tcs.ain.infoserver.servlet*

Class Diagrams

diagram servlet

Classes

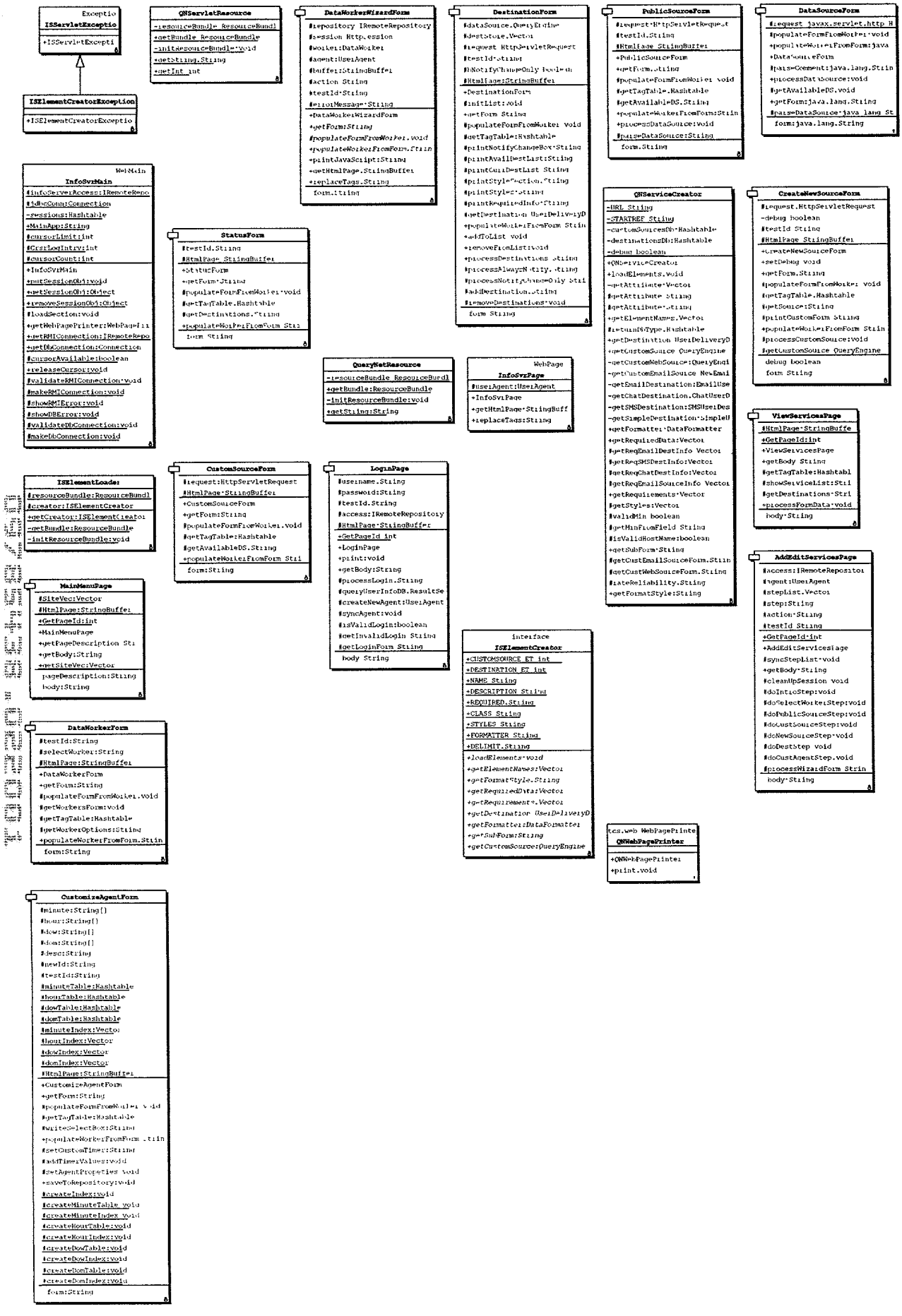
class tcs.ain.infoserver.servlet.AddEditServicesPage
class tcs.ain.infoserver.servlet.CreateNewSourceForm
class tcs.ain.infoserver.servlet.CustomSourceForm
class tcs.ain.infoserver.servlet.CustomizeAgentForm
class tcs.ain.infoserver.servlet.DataSourceForm
class tcs.ain.infoserver.servlet.DataWorkerForm
class tcs.ain.infoserver.servlet.DataWorkerWizardForm
class tcs.ain.infoserver.servlet.DestinationForm
class tcs.ain.infoserver.servlet.ISElementCreatorException
class tcs.ain.infoserver.servlet.ISElementLoader
class tcs.ain.infoserver.servlet.ISServletException
class tcs.ain.infoserver.servlet.InfoSvrMain
class tcs.ain.infoserver.servlet.InfoSvrPage
class tcs.ain.infoserver.servlet.LoginPage
class tcs.ain.infoserver.servlet.MainMenuPage
class tcs.ain.infoserver.servlet.PublicSourceForm
class tcs.ain.infoserver.servlet.QNServiceCreator
class tcs.ain.infoserver.servlet.QNServletResource
class tcs.ain.infoserver.servlet.QNWebPagePrinter
class tcs.ain.infoserver.servlet.QueryNetResource
class tcs.ain.infoserver.servlet.StatusForm
class tcs.ain.infoserver.servlet.ViewServicesPage

Interfaces

interface tcs.ain.infoserver.servlet.ISElementCreator

Class Diagrams

Class Diagram *servlet*


Exceptio
ISServletExceptio
+ISServletExcepti
ISElementCreatorException
+ISElementCreatorExceptio
QNServletResource
-resourceBundle ResourceBundl
+getBundle ResourceBundle
-initResourceBundle:void
+getString:String
+getInt int
DataWorkerWizardForm
#repository IRemoteRepository
#session Http.session
#worker:DataWorker
#agent:UserAgent
#buffer:StringBuffer
#action String
#testId:String
#errorMessage:String
+DataWorkerWizardForm
+getForm:String
#populateFormFromWorker.void
+populateWorkerFromForm.Strin
+printJavaScript:String
+getHtmlPage.StringBuffer
+replaceTags.String
form.String
DestinationForm
#dataSource.QueryEngine
#destStore.Vector
#request HttpServletRequest
#testId:String
#bNotifyChangeOnly boolean
#HtmlPage:StringBuffer
+DestinationForm
#initList:void
+getForm String
#populateFormFromWorker void
#getTagTable:Hashtable
#printNotifyChangeBox:String
#printAvailDestList:String
#printCurrDestList String
#printStyle.section.String
#printStyle:String
#printRequiredInfo:String
#getDestination UserDeliveryD
#populateWorkerFromForm Strin
+addToList void
+removeFromList:void
+processDestinations String
#processAlwaysNotify.String
#processNotifyChangeOnly Stri
#addDestination.String
#removeDestinations:void
form String
PublicSourceForm
#request:HttpServletRequest
#testId.String
#HtmlPage StringBuffer
+PublicSourceForm
+getForm.String
#populateFormFromWorker void
#getTagTable.Hashtable
#getAvailableDS.String
+populateWorkerFromForm:Strin
+processDataSource:void
#pairs:DataSource:String
form.String
DataSourceForm
#request javax.servlet.http H
#populateFormFromWorker:void
+populateWorkerFromForm:java
+DataSourceForm
#pairs:Comment:java.lang.Strin
+processDataSource:void
#getAvailableDS.void
+getForm:java.lang.String
#pairs:DataSource:java.lang.St
form:java.lang.String
WebMain
InfoSvrMain
#infoServerAccess:IRemoteRepo
#idbcConnConnection
-sessions:Hashtable
+MainApp:String
#cursorLimit:int
#CtrlNewEntry:int
#cursorCount:int
+InfoSvrMain
+putSessionObj:void
+getSessionObj:Object
+removeSessionObj:Object
#loadSection:void
+getWebPagePrinter:WebPageIn
+getRMIConnection:IRemoteRepo
+getDbConnection:Connection
#cursorAvailable:boolean
+releaseCursor:void
#validateRMIConnection:void
#makeRMIConnection:void
#showRMIError:void
#showDBError:void
#validateDbConnection:void
#makeDbConnection:void
StatusForm
#testId.String
#HtmlPage StringBuffer
+StatusForm
+getForm:String
#populateFormFromWorker:void
#getTagTable.Hashtable
#getDestinations.String
+populateWorkerFromForm Stri
form String
QNServiceCreator
-URL String
-STARTREF String
-customSourcesDb:Hashtable
-destinationsDb:Hashtable
-debug boolean
+QNServiceCreator
+loadElements.void
-getAttribute:Vector
#getAttribute String
#getAttribute:String
+getElementNames.Vector
#returnMyType.Hashtable
+getDestination UserDeliveryD
+getCustomSource QueryEngine
-getCustomWebSource:QueryEngi
-getCustomEmailSource NewEmai
-getEmailDestination:EmailUse
-getChatDestination.ChatUserD
-getSMSDestination:SMSUserDes
-getSimpleDestination:SimpleM
+getFormatter:DataFormatter
+getRequiredData:Vector
#getReqEmailDestInfo Vector
#getReqSMSDestInfo:Vector
#getReqChatDestInfo:Vector
#getReqEmailSourceInfo Vector
+getRequirements:Vector
#getStyles:Vector
#validMin boolean
#getMinFromField String
#isValidHostName:boolean
+getSubForm:String
#getCustEmailSourceForm.Strin
#getCustWebSourceForm.String
#rateReliability.String
+getFormatStyle:String
CreateNewSourceForm
#request.HttpServletRequest
-debug boolean
#testId String
#HtmlPage StringBuffer
+CreateNewSourceForm
+setDebug void
+getForm.String
#populateFormFromWorker void
#getTagTable.Hashtable
#getSource:String
#printCustomForm String
+populateWorkerFromForm Strin
#processCustomSource:void
#getCustomSource QueryEngine
debug boolean
form String
QueryNetResource
-resourceBundle ResourceBundl
+getBundle:ResourceBundle
-initResourceBundle:void
+getString:String
WebPage
InfoSvrPage
#userAgent:UserAgent
+InfoSvrPage
+getHtmlPage:StringBuff
+replaceTags:String
ViewServicesPage
#HtmlPage:StringBuffe
#GetPageId:int
+ViewServicesPage
+getBody String
#getTagTable:Hashtabl
#showServiceList:Stri
#getDestinations:Stri
+processFormData:void
body:String
ISElementLoade:
#resourceBundle:ResourceBundl
#creator:ISElementCreator
+getCreator:ISElementCreato
-getBundle:ResourceBundle
-initResourceBundle:void
CustomSourceForm
#request:HttpServletRequest
#HtmlPage:StringBuffer
+CustomSourceForm
+getForm:String
#populateFormFromWorker.void
#getTagTable:Hashtable
#getAvailableDS.String
+populateWorkerFromForm Stri
form:String
LoginPage
#username.String
#password:String
#testId.String
#access:IRemoteRepository
#HtmlPage:StringBuffer
+GetPageId int
+LoginPage
+print:void
+getBody:String
#processLogin.String
#queryUserInfoDB.ResultSe
#createNewAgent:UserAgent
#syncAgent:void
#isValidLogin:boolean
#getInvalidLogin String
#getLoginForm String
body String
AddEditServicesPage
#access:IRemoteReposito
#agent:UserAgent
#stepList.Vecto
#step:String
#action:String
#testId String
+GetPageId:int
+AddEditServicesPage
#syncStepList:void
+getBody:String
#cleanUpSession void
#doIntroStep:void
#doSelectWorkerStep:void
#doPublicSourceStep:void
#doCustSourceStep:void
#doNewSourceStep:void
#doDestStep void
#doCustAgentStep.void
#processWizardForm Strin
body:String
MainMenuPage
#SiteVec:Vector
#HtmlPage:StringBuffer
+GetPageId:int
+MainMenuPage
+getPageDescription Stri
+getBody:String
+getSiteVec:Vector
pageDescription:String
body:String
interface
ISElementCreator
+CUSTOMSOURCE_ET int
+DESTINATION_ET int
+NAME String
+DESCRIPTION String
+REQUIRED.String
+CLASS String
+STYLES String
+FORMATTER String
+DELIMIT.String
+loadElements:void
+getElementNames:Vector
+getFormatStyle.String
+getRequiredData:Vector
+getRequirements.Vector
+getDestination UserDeliveryD
+getFormatter:DataFormatter
+getSubForm:String
+getCustomSource:QueryEngine
DataWorkerForm
#testId:String
#selectWorker:String
#HtmlPage:StringBuffer
+DataWorkerForm
+getForm:String
#populateFormFromWorker.void
#getWorkersForm:void
#getTagTable:Hashtable
#getWorkerOptions:String
+populateWorkerFromForm.Strin
form:String
tcs.web WebPagePrinte
QNWebPagePrinter
+QNWebPagePrinter
+print.void
CustomizeAgentForm
#minute:String[]
#hour:String[]
#dow:String[]
#dom:String[]
#desc:String
#newId:String
#testId:String
#minuteTable:Hashtable
#hourTable:Hashtable
#dowTable:Hashtable
#domTable:Hashtable
#minuteIndex:Vector
#hourIndex:Vector
#dowIndex:Vector
#domIndex:Vector
#HtmlPage:StringBuffer
+CustomizeAgentForm
+getForm:String
#populateFormFromWorker void
#getTagTable:Hashtable
#writeSelectBox:String
+populateWorkerFromForm .strin
#setCustomTimer:String
#addTimerValues:void
#setAgentProperties void
+saveToRepository:void
#createIndex:void
#createMinuteTable void
#createMinuteIndex void
#createHourTable:void
#createHourIndex:void
#createDowTable:void
#createDowIndex:void
#createDomTable:void
#createDomIndex:void
form:String

Class Nodes

tcs.ain.infoserver.servlet.AddEditServicesPage
tcs.ain.infoserver.servlet.CreateNewSourceForm
tcs.ain.infoserver.servlet.CustomizeAgentForm
tcs.ain.infoserver.servlet.CustomSourceForm
tcs.ain.infoserver.servlet.DataSourceForm
tcs.ain.infoserver.servlet.DataWorkerForm
tcs.ain.infoserver.servlet.DataWorkerWizardForm
tcs.ain.infoserver.servlet.DestinationForm
tcs.ain.infoserver.servlet.InfoSvrMain
tcs.ain.infoserver.servlet.InfoSvrPage
tcs.ain.infoserver.servlet.ISElementCreatorException
tcs.ain.infoserver.servlet.ISElementLoader
tcs.ain.infoserver.servlet.ISServletException
tcs.ain.infoserver.servlet.LoginPage
tcs.ain.infoserver.servlet.MainMenuPage
tcs.ain.infoserver.servlet.PublicSourceForm
tcs.ain.infoserver.servlet.QNServiceCreator
tcs.ain.infoserver.servlet.QNServletResource
tcs.ain.infoserver.servlet.QNWebPagePrinter
tcs.ain.infoserver.servlet.QueryNetResource
tcs.ain.infoserver.servlet.StatusForm
tcs.ain.infoserver.servlet.ViewServicesPage

Interface Nodes

tcs.ain.infoserver.servlet.ISElementCreator

Class Node Detail

Class *tcs.ain.infoserver.servlet.AddEditServicesPage*

Extends:
InfoSvrPage

Generalization links

to **Class** InfoSvrPage

Class *tcs.ain.infoserver.servlet.CreateNewSourceForm*

Define the Data Sources that are desired.

Extends:
DataWorkerWizardForm

Generalization links

to **Class** DataWorkerWizardForm

Class *tcs.ain.infoserver.servlet.CustomizeAgentForm*

Define the Data Sources that are desired.

Extends:
DataWorkerWizardForm

Generalization links

to **Class** DataWorkerWizardForm

Class *tcs.ain.infoserver.servlet.CustomSourceForm*

Define the Data Sources that are desired.

Extends:
DataWorkerWizardForm

Generalization links

to **Class** DataWorkerWizardForm

Class *tcs.ain.infoserver.servlet.DataSourceForm*

Extends:
tcs.ain.infoserver.servlet.DataWorkerWizardForm

Generalization links

to **Class** DataWorkerWizardForm

Class *tcs.ain.infoserver.servlet.DataWorkerForm*

Define the Data Sources that are desired.

Extends:
DataWorkerWizardForm

Generalization links

to **Class** DataWorkerWizardForm

Class *tcs.ain.infoserver.servlet.DataWorkerWizardForm*

Base class for elements that will show info about a DataWorker Subclasses will access each component that a DataWorker cares about

Class *tcs.ain.infoserver.servlet.DestinationForm*

Define the Data Sources that are desired.

Extends:
DataWorkerWizardForm

Generalization links

to **Class** DataWorkerWizardForm

Class *tcs.ain.infoserver.servlet.InfoSvrMain*

This is the Main page for the Info Server Web Application. This loads pages according to the received Section variable

Extends:
WebMain

Class *tcs.ain.infoserver.servlet.InfoSvrPage*

Extends:
WebPage

Class *tcs.ain.infoserver.servlet.ISElementCreatorException*

Extends:
ISServletException

Generalization links

to **Class** ISServletException

Class *tcs.ain.infoserver.servlet.ISElementLoader*

Association links

to **Interface** ISElementCreator

Class *tcs.ain.infoserver.servlet.ISServletException*

Extends:
Exception

Class *tcs.ain.infoserver.servlet.LoginPage*

USER LOGIN PAGE. This prompts user for username/pw and tries and to find (or create) a UserAgent object for them. The UserAgent object is inserted into the user's Session under the name 'userAgent', for retention between screens This will eventually validate against and reference the SMS Internet Suite UserInfo database. Note: New agents are created for authenticated users, and their WebProv values will be assigned to the agent for name, email, min We will need to keep the agent info up-to-date somehow.

Extends:
InfoSvrPage

Generalization links

to **Class** InfoSvrPage

Class *tcs.ain.infoserver.servlet.MainMenuPage*

MAIN MENU PAGE. This will provide the main options for the app.

Extends:
InfoSvrPage

Generalization links

to **Class** InfoSvrPage

Class *tcs.ain.infoserver.servlet.PublicSourceForm*

Define the Data Sources that are desired.

Extends:
DataWorkerWizardForm

Generalization links

to **Class** DataWorkerWizardForm

Class *tcs.ain.infoserver.servlet.QNServiceCreator*

S. Levine on 11/1/99

Implements:
ISElementCreator

Implementation links

to **Interface** ISElementCreator

Class *tcs.ain.infoserver.servlet.QNServletResource*

This class is a singleton for a property resource bundle to be used by the Query Net Wizard. The resource bundle contains values that could potentially displayed to users. It also provides a direct, static method for safely getting String values without worrying about MIssingResourceExceptions ras; 2/29/99; modified by S. Levine on 9/9/99

Class *tcs.ain.infoserver.servlet.QNWebPagePrinter*

Extends:
tcs.web.WebPagePrinter

Class *tcs.ain.infoserver.servlet.QueryNetResource*

This class is a singleton for a property resource bundle to be used by the Query Net Wizard. The resource bundle contains values that could potentially displayed to users. It also provides a direct, static method for safely getting String values without worrying about MIssingResourceExceptions ras; 2/29/99; modified by S. Levine on 9/9/99

Class *tcs.ain.infoserver.servlet.StatusForm*

Define the Data Sources that are desired.

Extends:
DataWorkerWizardForm

Generalization links

to **Class** DataWorkerWizardForm

Class *tcs.ain.infoserver.servlet.ViewServicesPage*

Extends:
InfoSvrPage

Generalization links

to **Class** InfoSvrPage

Interface Node Detail

Interface *tcs.ain.infoserver.servlet.ISElementCreator*

Class Detail

Class *tcs.ain.infoserver.servlet.AddEditServicesPage*

```
public class AddEditServicesPage
```

Extends:
tcs.ain.infoserver.servlet.InfoSvrPage

Attributes

access
action
agent
step
stepList
testId

Constructors

AddEditServicesPage(HttpServletRequest, HttpServletResponse, IRemoteRepository, UserAgent)

Operations

cleanUpSession()
doCustAgentStep()
doCustSourceStep()
doDestStep()
doIntroStep()
doNewSourceStep()
doPublicSourceStep()
doSelectWorkerStep()
getBody()
GetPageId()
processWizardForm(DataWorkerWizardForm, boolean, boolean)
syncStepList(Vector, String)

Attribute Detail

access

```
protected IRemoteRepository access
```

action

```
protected String action
```

agent

```
protected UserAgent agent
```

step

```
protected String step = null
```

stepList

```
protected Vector stepList
```

testId

```
protected String testId
```

Constructor Detail

AddEditServicesPage

```
public AddEditServicesPage(HttpServletRequest request, HttpServletResponse response,
IRemoteRepository access, UserAgent agent)
```

Constructor calls super() and sets the title for this page

Operation Detail

cleanUpSession

```
protected void cleanUpSession()
```

doCustAgentStep

```
protected void doCustAgentStep()
```

doCustSourceStep

```
protected void doCustSourceStep()
```

doDestStep

```
protected void doDestStep()
```

doIntroStep

```
protected void doIntroStep()
```

doNewSourceStep

```
protected void doNewSourceStep()
```

doPublicSourceStep

```
protected void doPublicSourceStep()
```

doSelectWorkerStep

```
protected void doSelectWorkerStep()
```

getBody

```
public String getBody()
```

Return the page's body. For increased efficiency, this directly writes to the response output and does not actually return any data.

GetPageId

```
public static int GetPageId()
```

processWizardForm

```
protected String processWizardForm(DataWorkerWizardForm wizardForm, boolean doProcess, boolean
doPrint)
```

syncStepList

```
protected void syncStepList(Vector list, String currStep)
```

Return the page's body. For increased efficiency, this directly writes to the response output and does not actually return any data.

Class *tcs.ain.infoserver.servlet.CreateNewSourceForm*

```
public class CreateNewSourceForm
```

Extends:

tcs.ain.infoserver.servlet.DataWorkerWizardForm

Define the Data Sources that are desired.

Attributes

debug
HtmlPage
request
testId

Constructors

CreateNewSourceForm(HttpServletRequest, IRemoteRepository, String)

Operations

getCustomSource(String, StringBuffer)
getForm()
getSource()
getTagTable()
populateFormFromWorker()
populateWorkerFromForm()
printCustomForm(String)
processCustomSource(StringBuffer)
setDebug(boolean)

Attribute Detail

debug

```
private boolean debug = true
```

For Internal Debugging

HtmlPage

```
protected static StringBuffer HtmlPage
```

Stores Html Page associated with a form producing class

request

```
protected HttpServletRequest request
```

HttpServletRequest

testId

```
protected String testId
```

Stores error messages recieved from creating a new source

Constructor Detail

CreateNewSourceForm

```
public CreateNewSourceForm(HttpServletRequest request, IRemoteRepository access, String stepDesc)
```

Constructor

Operation Detail

getCustomSource

```
protected QueryEngine getCustomSource(String source, StringBuffer errorMsg)
```

Get a new Custom DataSource from the ISElementCreator Implementation

getForm

```
public String getForm()
```

Return the appropriate form for adding new workers or selecting existing workers

getSource

```
protected String getSource()
```

get selected source name from request object

getTagTable

```
protected Hashtable getTagTable()
```

Create a table of tags and their values to be used in replacing tags found in static html pages. Used to add dynamic content to these pages.

populateFormFromWorker

```
protected void populateFormFromWorker()
```

Populate the form fields with values from the DataWorker

populateWorkerFromForm

```
public String populateWorkerFromForm()
```

Populate the DataWorker with data assigned to the form

printCustomForm

```
protected String printCustomForm(String source)
```

processCustomSource

```
protected void processCustomSource(StringBuffer errorMsg)
```

Handle logic flow of creating a new custom datasource

setDebug

```
public void setDebug(boolean b)
```

Return the appropriate form for adding new workers or selecting existing workers

Class *tcs.ain.infoserver.servlet.CustomizeAgentForm*

```
public class CustomizeAgentForm
```

Extends:

tcs.ain.infoserver.servlet.DataWorkerWizardForm

Define the Data Sources that are desired.

Attributes

desc
dom
domIndex
domTable
dow
dowIndex
dowTable
hour
hourIndex
hourTable
HtmlPage
minute
minuteIndex
minuteTable
newId
testId

Constructors

CustomizeAgentForm(HttpServletRequest, IRemoteRepository, String)

Operations

addTimerValues(String[], Hashtable, TimeElement)
createDomIndex()
createDomTable()
createDowIndex()
createDowTable()
createHourIndex()
createHourTable()
createIndex(Vector, String)
createMinuteIndex()
createMinuteTable()
getForm()
getTagTable()
populateFormFromWorker()
populateWorkerFromForm()
saveToRepository()
setAgentPropeties()
setCustomTimer()
writeSelectBox(String, Hashtable, Vector, TimeElement)

Attribute Detail

desc

protected String desc

dom

protected String[] dom

domIndex

protected static Vector domIndex = null

domTable

protected static Hashtable domTable = null

dow

protected String[] dow

dowIndex

protected static Vector dowIndex = null

dowTable

protected static Hashtable dowTable = null

hour

protected String[] hour

hourIndex

protected static Vector hourIndex = null

hourTable

protected static Hashtable hourTable = null

HtmlPage

protected static StringBuffer HtmlPage

Stores Html Page associated with a form producing class

minute

protected String[] minute

minuteIndex

protected static Vector minuteIndex = null

minuteTable

protected static Hashtable minuteTable = null

newId

protected String newId

testId

protected String testId

Constructor Detail

CustomizeAgentForm

public CustomizeAgentForm(HttpServletRequest request, IRemoteRepository access, String stepDesc)

Constructor

Operation Detail

addTimerValues

```
protected void addTimerValues(String[] timerValues, Hashtable table, TimeElement timeElement)
```

createDomIndex

```
protected static void createDomIndex()
```

createDomTable

```
protected static void createDomTable()
```

Create a table to hold Day of Month options to be displayed to user as well as the corresponding TimeElement values

createDowIndex

```
protected static void createDowIndex()
```

createDowTable

```
protected static void createDowTable()
```

Create a table to hold Day of Week options to be displayed to user as well as the corresponding TimeElement values

createHourIndex

```
protected static void createHourIndex()
```

createHourTable

```
protected static void createHourTable()
```

Create a table to hold Hout options to be displayed to user as well as the corresponding TimeElement values

createIndex

```
protected static void createIndex(Vector vIndex, String property)
```

---------- /create vector list of possible index value for the specified index from the specified /resource string.

createMinuteIndex

```
protected static void createMinuteIndex()
```

createMinuteTable

```
protected static void createMinuteTable()
```

Create a table to hold Minute options to be displayed to user as well as the corresponding TimeElement values

getForm

```
public String getForm()
```

Return the appropriate form for adding new workers or selecting existing workers

getTagTable

```
protected Hashtable getTagTable()
```

Create a table of tags and their values to be used in replacing tags found in static html pages. Used to add dynamic content to these pages.

populateFormFromWorker

```
protected void populateFormFromWorker()
```

Populate the form fields with values from the DataWorker

populateWorkerFromForm

```
public String populateWorkerFromForm()
```

Populate the DataWorker with data assigned to the form

saveToRepository

```
public void saveToRepository()
```

setAgentPropeties

```
protected void setAgentPropeties()
```

setCustomTimer

```
protected String setCustomTimer()
```

writeSelectBox

```
protected String writeSelectBox(String name, Hashtable table, Vector index, TimeElement
timeElement)
```

Class *tcs.ain.infoserver.servlet.CustomSourceForm*

```
public class CustomSourceForm
```

Extends:

tcs.ain.infoserver.servlet.DataWorkerWizardForm

Define the Data Sources that are desired.

Attributes

HtmlPage
request

Constructors

CustomSourceForm(HttpServletRequest, IRemoteRepository, String)

Operations

getAvailableDS()
getForm()
getTagTable()
populateFormFromWorker()
populateWorkerFromForm()

Attribute Detail

HtmlPage

```
protected static StringBuffer HtmlPage
```

Stores Html Page associated with a form producing class

request

```
protected HttpServletRequest request
```

Constructor Detail

CustomSourceForm

```
public CustomSourceForm(HttpServletRequest request, IRemoteRepository access, String stepDesc)
```

Constructor

Operation Detail

getAvailableDS

```
protected String getAvailableDS()
```

create select box for available datasources and store in buffer

getForm

```
public String getForm()
```

Return the appropriate form for adding new workers or selecting existing workers

getTagTable

```
protected Hashtable getTagTable()
```

Create a table of tags and their values to be used in replacing tags found in static html pages. Used to add dynamic content to these pages.

populateFormFromWorker

```
protected void populateFormFromWorker()
```

Populate the form fields with values from the DataWorker

populateWorkerFromForm

```
public String populateWorkerFromForm()
```

Edit Data Worker based on user request

Class *tcs.ain.infoserver.servlet.DataSourceForm*

```
public class DataSourceForm
```

Extends:

tcs.ain.infoserver.servlet.DataWorkerWizardForm

Attributes

request

Constructors

DataSourceForm(javax.servlet.http.HttpServletRequest, tcs.ain.infoserver.access.IRemoteRepository, java.lang.String)

Operations

getAvailableDS()
getForm()
parseComment(java.lang.String)
parseDataSource(java.lang.String)
populateFormFromWorker()
populateWorkerFromForm()
processDataSource()

Attribute Detail

request

```
protected javax.servlet.http.HttpServletRequest request
```

Constructor Detail

DataSourceForm

```
public DataSourceForm(javax.servlet.http.HttpServletRequest,
tcs.ain.infoserver.access.IRemoteRepository, java.lang.String)
```

Operation Detail

getAvailableDS

```
protected void getAvailableDS()
```

getForm

```
public java.lang.String getForm()
```

parseComment

```
protected java.lang.String parseComment(java.lang.String)
```

parseDataSource

```
protected java.lang.String parseDataSource(java.lang.String)
```

populateFormFromWorker

```
protected void populateFormFromWorker()
```

populateWorkerFromForm

```
public java.lang.String populateWorkerFromForm()
```

processDataSource

```
public void processDataSource()
```

Class *tcs.ain.infoserver.servlet.DataWorkerForm*

```
public class DataWorkerForm
```

Extends:

tcs.ain.infoserver.servlet.DataWorkerWizardForm

Define the Data Sources that are desired.

Attributes

HtmlPage
selectWorker
testId

Constructors

DataWorkerForm(HttpServletRequest, IRemoteRepository, String)

Operations

getForm()
getTagTable()
getWorkerOptions()
getWorkersForm()
populateFormFromWorker()
populateWorkerFromForm()

Attribute Detail

HtmlPage

```
protected static StringBuffer HtmlPage
```

Stores Html Page associated with a form producing class

selectWorker

```
protected String selectWorker
```

Name of Selected worker to be edited

testId

```
protected String testId
```

Id to be used for script testing only

Constructor Detail

DataWorkerForm

```
public DataWorkerForm(HttpServletRequest request, IRemoteRepository access, String stepDesc)
```

Constructor

Operation Detail

getForm

```
public String getForm()
```

Return the appropriate form for adding new workers or selecting existing workers

getTagTable

```
protected Hashtable getTagTable()
```

Create a table of tags and their values to be used in replacing tags found in static html pages. Used to add dynamic content to these pages.

getWorkerOptions

```
protected String getWorkerOptions()
```

getWorkersForm

```
protected void getWorkersForm()
```

Create an html form for selecting or removing existing Data Workers. This is the initial form for the Query Net Wizard

populateFormFromWorker

```
protected void populateFormFromWorker()
```

Populate the form fields with values from the DataWorker

populateWorkerFromForm

```
public String populateWorkerFromForm()
```

Populate the DataWorker with data assigned to the form

Class *tcs.ain.infoserver.servlet.DataWorkerWizardForm*

```
public abstract class DataWorkerWizardForm
```

Base class for elements that will show info about a DataWorker Subclasses will access each component that a DataWorker cares about

Attributes

action
agent
buffer
errorMessage
repository
session
testId
worker

Constructors

DataWorkerWizardForm(HttpServletRequest, IRemoteRepository, String)

Operations

getForm()
getHtmlPage(String)
populateFormFromWorker()
populateWorkerFromForm()
printJavaScript()
replaceTags(String, Hashtable)

Attribute Detail

action

```
protected String action
```

agent

```
protected UserAgent agent
```

buffer

```
protected StringBuffer buffer
```

errorMessage

```
protected String errorMessage = ""
```

repository

```
protected IRemoteRepository repository
```

session

```
protected HttpSession session
```

testId

```
protected String testId
```

Used for testing scripts only

worker

```
protected DataWorker worker
```

Constructor Detail

DataWorkerWizardForm

```
public DataWorkerWizardForm(HttpServletRequest request, IRemoteRepository r, String errorMessage)
```

Constructor

Operation Detail

getForm

```
public abstract String getForm()
```

Return the HTML string that defines the form

getHtmlPage

```
public StringBuffer getHtmlPage(String path)
```

Read in specified Html Page and store to static HtmlPage StringBuffer

Throws:

Exception

populateFormFromWorker

```
protected abstract void populateFormFromWorker()
```

Populate the form fields with values from the DataWorker

populateWorkerFromForm

```
public abstract String populateWorkerFromForm()
```

Populate the DataWorker with data assigned to the form

printJavaScript

```
public String printJavaScript()
```

java script for create pop up browser for displaying help info

replaceTags

```
public String replaceTags(String content, Hashtable ht)
```

Read in specified Html Page and store content to StringBuffer

Throws:

Exception

Class *tcs.ain.infoserver.servlet.DestinationForm*

```
public class DestinationForm
```

Extends:

tcs.ain.infoserver.servlet.DataWorkerWizardForm

Define the Data Sources that are desired.

Attributes

bNotifyChangeOnly
dataSource
destStore
HtmlPage
request
testId

Constructors

DestinationForm(HttpServletRequest, IRemoteRepository, String)

Operations

addDestination(String, IDataEventSource)
addToList()
getDestination(String)
getForm()
getTagTable()
initList()
populateFormFromWorker()
populateWorkerFromForm()
printAvailDestList()
printCurrDestList()
printNotifyChangeBox()
printRequiredInfo(String)
printStyles(String)
printStyleSection()
processAlwaysNotify()
processDestinations()
processNotifyChangeOnly()
removeDestinations(IDataEventSource)
removeFromList()

Attribute Detail

bNotifyChangeOnly

```
protected boolean bNotifyChangeOnly = false
```

dataSource

```
protected QueryEngine dataSource = null
```

destStore

```
protected Vector destStore = null
```

HtmlPage

```
protected static StringBuffer HtmlPage
```

Stores Html Page associated with a form producing class

request

```
protected HttpServletRequest request = null
```

testId

```
protected String testId = null
```

Constructor Detail

DestinationForm

```
public DestinationForm(HttpServletRequest request, IRemoteRepository access, String stepDesc)
```

Constructor

Operation Detail

addDestination

```
protected String addDestination(String destination, IDataEventSource iDes)
```

Add a destination to Data Worker

addToList

```
public void addToList()
```

Add new destinations to destStore Vector and store into session obj

getDestination

```
protected UserDeliveryDestination getDestination(String destId)
```

Return requested destination.

getForm

```
public String getForm()
```

Return the appropriate form for adding new workers or selecting existing workers

getTagTable

```
protected Hashtable getTagTable()
```

Create a table of tags and their values to be used in replacing tags found in static html pages. Used to add dynamic content to these pages.

initList

```
protected void initList()
```

Initialize destStore

populateFormFromWorker

```
protected void populateFormFromWorker()
```

Populate the form fields with values from the DataWorker

populateWorkerFromForm

```
public String populateWorkerFromForm()
```

Populate the DataWorker with data assigned to the form, or temporarily store.

printAvailDestList

```
protected String printAvailDestList()
```

Create an html Select box and populate with available destinations

printCurrDestList

```
protected String printCurrDestList()
```

Create an html Select box and populate with currently selected destinations

printNotifyChangeBox

```
protected String printNotifyChangeBox()
```

Create an html CheckBox for "Notify If Change Only" option

printRequiredInfo

```
protected String printRequiredInfo(String destId)
```

Create a text box and prompt for required info for a given destination

printStyles

```
protected String printStyles(String destId)
```

Create and populate a drop down box with available style options for a give data sources

printStyleSection

```
protected String printStyleSection()
```

Create table rows dynamically based on Data Source info

processAlwaysNotify

```
protected String processAlwaysNotify()
```

Add all destinations to Data Worker

processDestinations

```
public String processDestinations()
```

process selected destinations *

processNotifyChangeOnly

```
protected String processNotifyChangeOnly()
```

Add destinations to NotifyChangeOnlyDestination, then add NotifyChangeOnlyDestination to Worker

removeDestinations

```
protected void removeDestinations(IDataEventSource des)
```

Remove destination from Data Worker

removeFromList

```
public void removeFromList()
```

Remove selected destinations from destStore Vector

Class *tcs.ain.infoserver.servlet.InfoSvrMain*

```
public class InfoSvrMain
```

Extends:

tcs.web.WebMain

This is the Main page for the Info Server Web Application. This loads pages according to the received Section variable

Attributes

CrsrLogIntrv
cursorCount
cursorLimit
infoServerAccess
jdbcConn
MainApp
sessions

Constructors

InfoSvrMain()

Operations

cursorAvailable()
getDbConnection()
getRMIConnection()
getSessionObj(Object)
getWebPagePrinter(WebPage)
loadSection(HttpServletRequest, HttpServletResponse)
makeDbConnection()
makeRMIConnection()
putSessionObj(Object, Object)
releaseCursor()
removeSessionObj(Object)
showDBError(HttpServletResponse)
showRMIError(HttpServletResponse)
validateDbConnection()
validateRMIConnection()

Attribute Detail

CrsrLogIntrv

```
protected static final int CrsrLogIntrv = QNServletResource.getInt("DbCursorLogIntrv")
```

Amount of tries for a cursor to do before sending "WAITING FOR CURSOR" log message to stdout.

cursorCount

```
protected static int cursorCount = 0
```

The database cursors currently open by repository.

cursorLimit

```
protected static final int cursorLimit = QNServletResource.getInt("DbCursorLimit")
```

The database cursor limit for the Repository database connection.

infoServerAccess

```
protected static IRemoteRepository infoServerAccess = null
```

Remote InfoServer Access Repository for loading / saving User Agents and Sources. Accessed via RMI

jdbcConn

```
protected static Connection jdbcConn
```

JDBC Connection for querying provisioning information from database

MainApp

```
public static String MainApp = QNServletResource.getString("MainAppPath")
```

sessions

```
private static Hashtable sessions
```

used for script load testing where regular session objects are not available

Constructor Detail

InfoSvrMain

```
public InfoSvrMain()
```

Operation Detail

cursorAvailable

```
protected static boolean cursorAvailable()
```

Check to see if cursor limit has been reached

getDbConnection

```
public static Connection getDbConnection()
```

Return a reference to jdbcConn if the CursorLimit has not been exceeded and a valid connection can be established, otherwise return null. This protects from exceeding the a given Database Implementation's cursor limit setting. It is the calling procedures responsibility to call release cursor when finished using the jdbcConn reference. Throws exception if unable to establish a valid database connection

Throws:

SQLException

getRMIConnection

```
public static IRemoteRepository getRMIConnection()
```

Connect the remote object to the server if necessary and return RMI Connection

Throws:

Exception

getSessionObj

```
public static Object getSessionObj(Object key)
```

Use this instead of session object when script testing

getWebPagePrinter

```
public WebPagePrinter getWebPagePrinter(WebPage wp)
```

Connect the remote object to the server if necessary and return RMI Connection

Throws:

Exception

loadSection

```
protected void loadSection(HttpServletRequest request, HttpServletResponse response)
```

Evaluate the 'Section' value and load the appropriate page. If Section is not set, then just load the MONTHVIEW_PAGE page

makeDbConnection

```
protected static void makeDbConnection()
```

Make sure jdbcConn is valid. If not, report the error to out and return false

Throws:
SQLException

makeRMIConnection

```
protected static void makeRMIConnection()
```

Connect the remote object to the server if necessary

Throws:
Exception

putSessionObj

```
public static void putSessionObj(Object key, Object value)
```

Use this instead of session object when script testing

releaseCursor

```
public static void releaseCursor()
```

Decrement cursorCount so that another process may reference jdbcConn

removeSessionObj

```
public static Object removeSessionObj(Object key)
```

Use this instead of session object when script testing

showDBError

```
protected static void showDBError(HttpServletResponse response)
```

Display a spiffy RMI error message

showRMIError

```
protected static void showRMIError(HttpServletResponse response)
```

Display a spiffy RMI error message

validateDbConnection

```
protected static void validateDbConnection()
```

Make sure con is valid. If not, report the error to out and throw the exception

Throws:
SQLException

validateRMIConnection

```
protected static void validateRMIConnection()
```

make connection if necessary; test the connection; if failed, try to to remake the connection return true if good connection

Class *tcs.ain.infoserver.servlet.InfoSvrPage*

```
public abstract class InfoSvrPage
```

Extends:
tcs.web.WebPage

Attributes

userAgent

Constructors

InfoSvrPage(HttpServletRequest, HttpServletResponse, UserAgent)

Operations

getHtmlPage(String)
replaceTags(String, Hashtable)

Attribute Detail

userAgent

```
protected UserAgent userAgent
```

Constructor Detail

InfoSvrPage

```
public InfoSvrPage(HttpServletRequest req, HttpServletResponse res, UserAgent agent)
```

Stores Html Page associated with a form producing class

Operation Detail

getHtmlPage

```
public StringBuffer getHtmlPage(String path)
```

Read in specified Html Page and store to static HtmlPage StringBuffer

Throws:
Exception

replaceTags

```
public String replaceTags(String content, Hashtable ht)
```

Read in specified Html Page and store content to StringBuffer

Throws:
Exception

Class *tcs.ain.infoserver.servlet.ISElementCreatorException*

```
public class ISElementCreatorException
```

Extends:
tcs.ain.infoserver.servlet.ISServletException

Constructors

ISElementCreatorException(String)

Constructor Detail

ISElementCreatorException

```
public ISElementCreatorException(String s)
```

Class *tcs.ain.infoserver.servlet.ISElementLoader*

```
public class ISElementLoader
```

Attributes

creator

resourceBundle

Operations

getBundle()
getCreator()
initResourceBundle()

Attribute Detail

creator

```
protected static ISElementCreator creator = null
```

resourceBundle

```
protected static ResourceBundle resourceBundle = null
```

Operation Detail

getBundle

```
private static ResourceBundle getBundle()
```

Return the single, static instance of the resource bundle for the SMPP Module

getCreator

```
public static ISElementCreator getCreator()
```

Throws:

ClassNotFoundException
InstantiationException
IllegalAccessException

initResourceBundle

```
private static void initResourceBundle()
```

Create the resource bundle

Class *tcs.ain.infoserver.servlet.ISServletException*

```
public class ISServletException
```

Extends:

java.lang.Exception

Constructors

ISServletException(String)

Constructor Detail

ISServletException

```
public ISServletException(String s)
```

Class *tcs.ain.infoserver.servlet.LoginPage*

```
public class LoginPage
```

Extends:

tcs.ain.infoserver.servlet.InfoSvrPage

USER LOGIN PAGE. This prompts user for username/pw and tries and to find (or create) a UserAgent object for them. The UserAgent object is inserted into the user's Session under the name 'userAgent', for retention between screens This will eventually validate against and reference the SMS Internet Suite UserInfo database. Note: New agents are created for authenticated users, and their WebProv values will be assigned to the agent for name, email, min We will need to keep the agent info up-to-date somehow.

Attributes

access
HtmlPage
password
testId
username

Constructors

LoginPage(HttpServletRequest, HttpServletResponse, IRemoteRepository)

Operations

createNewAgent(ResultSet)
getBody()
getInvalidLogin()
getLoginForm()
GetPageId()
isValidLogin(ResultSet)
print()
processLogin()
queryUserInfoDB()
syncAgent(ResultSet, UserAgent)

Attribute Detail

access

```
protected IRemoteRepository access
```

How we obtain Agents

HtmlPage

```
protected static StringBuffer HtmlPage
```

Stores Html Page associated with a form producing class

password

```
protected String password
```

testId

```
protected String testId
```

username

```
protected String username
```

Constructor Detail

LoginPage

```
public LoginPage(HttpServletRequest request, HttpServletResponse response, IRemoteRepository
access)
```

Constructor calls super() and sets the title for this page

Operation Detail

createNewAgent

```
protected UserAgent createNewAgent(ResultSet r)
```

Create a new UserAgent based on information we can get from the UserProvisioning database for the login name

getBody

```
public String getBody()
```

Return the page's body

getInvalidLogin

```
protected String getInvalidLogin()
```

Return String describing that the login/pw were invalid

getLoginForm

```
protected String getLoginForm()
```

Show the login form. When accepted, this page will get called again, but then it will process the results

GetPageId

```
public static int GetPageId()
```

isValidLogin

```
protected boolean isValidLogin(ResultSet r)
```

Autherticate the user.

print

```
public void print()
```

Ensure that pagePrint is set to not print banner

processLogin

```
protected String processLogin()
```

process the login; if acceptable, stick in the Session and show Welcome Page

queryUserInfoDB

```
protected ResultSet queryUserInfoDB()
```

Query the UserProvisioning database for the login name

syncAgent

```
protected void syncAgent(ResultSet r, UserAgent agent)
```

Make sure agent info is up to date with provisioning information.

Class *tcs.ain.infoserver.servlet.MainMenuPage*

```
public class MainMenuPage
```

Extends:

tcs.ain.infoserver.servlet.InfoSvrPage

MAIN MENU PAGE. This will provide the main options for the app.

Attributes

HtmlPage
SiteVec

Constructors

MainMenuPage(HttpServletRequest, HttpServletResponse, UserAgent)

Operations

getBody()
getPageDescription()
GetPageId()
getSiteVec()

Attribute Detail

HtmlPage

```
protected static StringBuffer HtmlPage
```

Stores Html Page associated with a form producing class

SiteVec

```
protected static Vector SiteVec = new Vector()
```

This contains a central list of the site account names. Referenced elsewhere, but just populated here

Constructor Detail

MainMenuPage

```
public MainMenuPage(HttpServletRequest request, HttpServletResponse response, UserAgent agent)
```

Constructor calls super() and sets the title for this page

Operation Detail

getBody

```
public String getBody()
```

the page body

getPageDescription

```
public String getPageDescription()
```

This returns the special page description.

GetPageId

```
public static int GetPageId()
```

getSiteVec

```
public static Vector getSiteVec()
```

Return the static vector of available sites

Class *tcs.ain.infoserver.servlet.PublicSourceForm*

```
public class PublicSourceForm
```

Extends:

tcs.ain.infoserver.servlet.DataWorkerWizardForm

Define the Data Sources that are desired.

Attributes

HtmlPage
request
testId

Constructors

PublicSourceForm(HttpServletRequest, IRemoteRepository, String)

Operations

getAvailableDS()
getForm()
getTagTable()
parseDataSource(String)
populateFormFromWorker()

populateWorkerFromForm()
processDataSource()

Attribute Detail

HtmlPage

```
protected static StringBuffer HtmlPage
```

Stores Html Page associated with a form producing class

request

```
protected HttpServletRequest request
```

Request object

testId

```
protected String testId
```

Used for testing via scripts only

Constructor Detail

PublicSourceForm

```
public PublicSourceForm(HttpServletRequest request, IRemoteRepository access, String stepDesc)
```

Constructor

Operation Detail

getAvailableDS

```
protected String getAvailableDS()
```

create select box for available datasources and store in buffer

getForm

```
public String getForm()
```

Return the appropriate form for adding new workers or selecting existing workers

getTagTable

```
protected Hashtable getTagTable()
```

Create a table of tags and their values to be used in replacing tags found in static html pages. Used to add dynamic content to these pages.

parseDataSource

```
protected String parseDataSource(String source)
```

Remove comment from destination. Comment is delimited with ":" in front. ex./ parseDestination("EMAIL: This is a public email destination.") returns "EMAIL"

populateFormFromWorker

```
protected void populateFormFromWorker()
```

Populate the form fields with values from the DataWorker

populateWorkerFromForm

```
public String populateWorkerFromForm()
```

Edit Data Worker based on user request

processDataSource

```
public void processDataSource()
```

Edit Data Worker based on user request

Class *tcs.ain.infoserver.servlet.QNServiceCreator*

```
public class QNServiceCreator
```

Implements:

tcs.ain.infoserver.servlet.ISElementCreator

S. Levine on 11/1/99

Attributes

customSourcesDb
debug
destinationsDb
STARTREF
URL

Constructors

QNServiceCreator()

Operations

getAttribute(String, String, int)
getAttribute(String, String)
getAttribute(String, int, String)
getChatDestination(String, Vector, StringBuffer)
getCustEmailSourceForm(String, InfoServerElement, UserAgent)
getCustomEmailSource(String, HttpServletRequest, StringBuffer)
getCustomSource(String, HttpServletRequest, StringBuffer)
getCustomWebSource(String, HttpServletRequest, StringBuffer)
getCustWebSourceForm(String, InfoServerElement, UserAgent)
getDestination(String, Vector, StringBuffer)
getElementNames(int, boolean)
getEmailDestination(String, Vector, StringBuffer)
getFormatStyle(UserDeliveryDestination)
getFormatter(QueryEngine, String)
getMinFromField(String, String)
getReqChatDestInfo(InfoServerElement, UserAgent)
getReqEmailDestInfo(InfoServerElement, UserAgent)
getReqEmailSourceInfo(InfoServerElement, UserAgent)
getReqSMSDestInfo(InfoServerElement, UserAgent)
getRequiredData(String, InfoServerElement, UserAgent, int)
getRequirements(String, int)
getSimpleDestination(String, StringBuffer)
getSMSDestination(String, Vector, StringBuffer)
getStyles(String, int)
getSubForm(String, InfoServerElement, UserAgent, int)
isValidHostName(String)
loadElements(String, String)
rateReliability(String)
returnDbType(int)
validMin(String)

Attribute Detail

customSourcesDb

```
private Hashtable customSourcesDb = null
```

Store custom source elements.

debug

```
private static final boolean debug = true
```

For Internal Debugging

destinationsDb

```
private Hashtable destinationsDb = null
```

Store Destination elements.

STARTREF

```
private static final String STARTREF = QNConst.EDT_PREFIX + "STARTREF"
```

Name of url field Create Web Source Form

URL

```
private static final String URL = QNConst.EDT_PREFIX + "URL"
```

Name of url field Create Web Source Form

Constructor Detail

QNServiceCreator

```
public QNServiceCreator()
```

Return the single, static instance of the resource bundle for the SMPP Module

Operation Detail

getAttribute

```
protected String getAttribute(String name, String attr, int elementType)
```

Return the attribute from an Element Config String given the element name, attribute, and elementType.

getAttribute

```
protected String getAttribute(String configString, String atName)
```

Return the attribute from an Element Config String given the string, and the attribute.

getAttribute

```
private Vector getAttribute(String name, int elementType, String attr)
```

Return Parsed Attribute String, and return elements in a vector

getChatDestination

```
private ChatUserDestination getChatDestination(String destination, Vector reqInfo, StringBuffer
errorMsg)
```

Load elements descriptions from resource file into hashtables

getCustEmailSourceForm

```
protected String getCustEmailSourceForm(String name, InfoServerElement element, UserAgent agent)
```

Get form components necessary in creating the specifed element

getCustomEmailSource

```
private NewEmailQueryEngine getCustomEmailSource(String source, HttpServletRequest request,
StringBuffer errorMsg)
```

Load elements descriptions from resource file into hashtables

getCustomSource

```
public QueryEngine getCustomSource(String source, HttpServletRequest request, StringBuffer
errorMsg)
```

Load elements descriptions from resource file into hashtables

getCustomWebSource

```
private QueryEngine getCustomWebSource(String source, HttpServletRequest request, StringBuffer errorMsg)
```

Load elements descriptions from resource file into hashtables

getCustWebSourceForm

```
protected String getCustWebSourceForm(String name, InfoServerElement element, UserAgent agent)
```

Get form components necessary in creating a web datasource

getDestination

```
public UserDeliveryDestination getDestination(String destination, Vector reqInfo, StringBuffer errorMsg)
```

Load elements descriptions from resource file into hashtables

getElementNames

```
public Vector getElementNames(int elementType, boolean addDesc)
```

return list of elements descriptions from specified hashtable

getEmailDestination

```
private EmailUserDestination getEmailDestination(String destination, Vector reqInfo, StringBuffer errorMsg)
```

Load elements descriptions from resource file into hashtables

getFormatStyle

```
public String getFormatStyle(UserDeliveryDestination dest)
```

Return the name of the XSL format being used by this destination

getFormatter

```
public DataFormatter getFormatter(QueryEngine ds, String style)
```

Load elements descriptions from resource file into hashtables

getMinFromField

```
protected String getMinFromField(String value, String srchFld)
```

try to lookup and return the MIN from the UserInfo table using the recvd search field returns null if not found

getReqChatDestInfo

```
protected Vector getReqChatDestInfo(InfoServerElement element, UserAgent agent)
```

getReqEmailDestInfo

```
protected Vector getReqEmailDestInfo(InfoServerElement element, UserAgent agent)
```

getReqEmailSourceInfo

```
protected Vector getReqEmailSourceInfo(InfoServerElement element, UserAgent agent)
```

getReqSMSDestInfo

```
protected Vector getReqSMSDestInfo(InfoServerElement element, UserAgent agent)
```

getRequiredData

```
public Vector getRequiredData(String destId, InfoServerElement element, UserAgent agent, int elementType)
```

Return existing Required Data associated with an existing element

getRequirements

```
public Vector getRequirements(String name, int elementType)
```

Return Required Information necessary for creating an element of this type

getSimpleDestination

```
private SimpleUserDestination getSimpleDestination(String destination, StringBuffer errorMsg)
```

Create a Simple user destination

getSMSDestination

```
private SMSUserDestination getSMSDestination(String destination, Vector reqInfo, StringBuffer errorMsg)
```

Load elements descriptions from resource file into hashtables

getStyles

```
protected Vector getStyles(String name, int elementType)
```

Return Required Information necessary for creating an element of this type

getSubForm

```
public String getSubForm(String name, InfoServerElement element, UserAgent agent, int elementType)
```

Get form components necessary in creating the specifed element

isValidHostName

```
protected boolean isValidHostName(String host)
```

Check to see if host name is valid. Return True if valid False if invalid.

loadElements

```
public void loadElements(String sourceElements, String destElements)
```

Load elements descriptions into hashtables, this method should only ever be called once by the loader so no need to synchronize.

rateReliability

```
protected String rateReliability(String key)
```

Relate reliability of Xml parsing key based on a scale of 1-10 were 1=most reliable and 10=least reliable. The less tags in the key, the more reliable. 4 tags or less is a 1, and every tag after that adds 1 to the rating. For ex./ Clip.Html[0].Body[0].Table[0] is very reliable and would be a 1, and Clip.Html[0].Body[0].Table[2].TR[1].TD[2] would be less reliable(but still reasonably reliable) and would be a 3

returnDbType

```
protected Hashtable returnDbType(int elementType)
```

Return correct hashtable dependent on elementType

validMin

```
protected boolean validMin(String address)
```

Look at the msgMIN field to ensure that it really contains a MIN. It is possible that it could actually contain alternate addressing info, which must be translated to a MIN address.

Class *tcs.ain.infoserver.servlet.QNServletResource*

```
public class QNServletResource
```

This class is a singleton for a property resource bundle to be used by the Query Net Wizard. The resource bundle contains values that could potentially displayed to users. It also provides a direct, static method for safely getting String values without worrying about MIssingResourceExceptions ras; 2/29/99; modified by S. Levine on 9/9/99

Attributes

resourceBundle

Operations

getBundle()
getInt(String)
getString(String)
initResourceBundle()

Attribute Detail

resourceBundle

```
private static ResourceBundle resourceBundle = null
```

The resource bundle we keep for use by the package.

Operation Detail

getBundle

```
public static ResourceBundle getBundle()
```

Return the single, static instance of the resource bundle for the SMPP Module

getInt

```
public static int getInt(String key)
```

Pass through to the resourceBundle, except that we internally catch the MissingResourceException and report the error but return a "?". Return Int value.

getString

```
public static String getString(String key)
```

Pass through to the resourceBundle, except that we internally catch the MissingResourceException and report the error but return a "?"

initResourceBundle

```
private static void initResourceBundle()
```

Create the resource bundle

Class *tcs.ain.infoserver.servlet.QNWebPagePrinter*

```
public class QNWebPagePrinter
```

Extends:

tcs.web.WebPagePrinter

Constructors

QNWebPagePrinter(tcs.web.WebPage)

Operations

print()

Constructor Detail

QNWebPagePrinter

```
public QNWebPagePrinter(tcs.web.WebPage)
```

Operation Detail

print

```
public void print()
```

Class *tcs.ain.infoserver.servlet.QueryNetResource*

```
class QueryNetResource
```

This class is a singleton for a property resource bundle to be used by the Query Net Wizard. The resource bundle contains values that could potentially displayed to users. It also provides a direct, static method for safely getting String values without worrying about MIssingResourceExceptions ras; 2/29/99; modified by S. Levine on 9/9/99

Attributes

resourceBundle

Operations

getBundle()
getString(String)
initResourceBundle()

Attribute Detail

resourceBundle

```
private static ResourceBundle resourceBundle = null
```

The resource bundle we keep for use by the package.

Operation Detail

getBundle

```
public static ResourceBundle getBundle()
```

Return the single, static instance of the resource bundle for the SMPP Module

getString

```
public static String getString(String key)
```

Pass through to the resourceBundle, except that we internally catch the MissingResourceException and report the error but return a "?"

initResourceBundle

```
private static void initResourceBundle()
```

Create the resource bundle

Class *tcs.ain.infoserver.servlet.StatusForm*

```
public class StatusForm
```

Extends:

tcs.ain.infoserver.servlet.DataWorkerWizardForm

Define the Data Sources that are desired.

Attributes

HtmlPage
testId

Constructors

StatusForm(HttpServletRequest, IRemoteRepository, String)

Operations

getDestinations()
getForm()
getTagTable()
populateFormFromWorker()
populateWorkerFromForm()

Attribute Detail

HtmlPage

```
protected static StringBuffer HtmlPage
```

Stores Html Page associated with a form producing class

testId

```
protected String testId
```

Id to be used for script testing only

Constructor Detail

StatusForm

```
public StatusForm(HttpServletRequest request, IRemoteRepository access, String stepDesc)
```

Constructor

Operation Detail

getDestinations

```
protected String getDestinations()
```

Return destinations associated with dataworker in html tag IMPORTANT: 1.) ONLY FILTERS WILL HAVE CHILD LISTENERS 2.) CHILD LISTENERS CAN BE OTHER FILTERS OR DESTINATIONS 3.) CHILD LISTENERS WILL NOT HAVE SIBLINGS 4.) DESTINATIONS WILL NOT HAVE CHILD LISTENERS ex/ ChangeOnlyFilter | SearchFilter | | | SMS Dest. Email Dest. Chat Dest.

getForm

```
public String getForm()
```

Return the appropriate form for adding new workers or selecting existing workers

getTagTable

```
protected Hashtable getTagTable()
```

Create a table of tags and their values to be used in replacing tags found in static html pages. Used to add dynamic content to these pages.

populateFormFromWorker

```
protected void populateFormFromWorker()
```

Populate the form fields with values from the DataWorker

populateWorkerFromForm

```
public String populateWorkerFromForm()
```

Populate the DataWorker with data assigned to the form

Class *tcs.ain.infoserver.servlet.ViewServicesPage*

```
public class ViewServicesPage
```

Extends:

tcs.ain.infoserver.servlet InfoSvrPage

Attributes

HtmlPage

Constructors

ViewServicesPage(HttpServletRequest, HttpServletResponse, UserAgent)

Operations

getBody()
getDestinations(DataWorker)
GetPageId()
getTagTable()
processFormData(HttpServletRequest)
showServiceList()

Attribute Detail

HtmlPage

```
protected static StringBuffer HtmlPage
```

Stores Html Page associated with a form producing class

Constructor Detail

ViewServicesPage

```
public ViewServicesPage(HttpServletRequest request, HttpServletResponse response, UserAgent agent)
```

Constructor calls super() and sets the title for this page

Operation Detail

getBody

```
public String getBody()
```

Return the page's body. For increased efficiency, this directly writes to the response output and does not actually return any data.

getDestinations

```
protected String getDestinations(DataWorker worker)
```

Return destinations associated with dataworker in html tag IMPORTANT: 1.) ONLY FILTERS WILL HAVE CHILD LISTENERS 2.) CHILD LISTENERS CAN BE OTHER FILTERS OR DESTINATIONS 3.) CHILD LISTENERS WILL NOT HAVE SIBLINGS 4.) DESTINATIONS WILL NOT HAVE CHILD LISTENERS ex/ ChangeOnlyFilter | SearchFilter | | | SMS Dest. Email Dest. Chat Dest.

GetPageId

```
public static int GetPageId()
```

getTagTable

```
protected Hashtable getTagTable()
```

Create a table of tags and their values to be used in replacing tags found in static html pages. Used to add dynamic content to these pages.

processFormData

```
public void processFormData(HttpServletRequest request)
```

Start and stop workers based on user input

showServiceList

```
protected String showServiceList()
```

Create a table and show a summary of each of the services (ie, Workers owned by the userAgent. The worker titles will also contain hyperlins for more information and editing choices

Interface Detail

Interface *tcs.ain.infoserver.servlet.ISElementCreator*

```
public interface ISElementCreator
```

Attributes

CLASS
CUSTOMSOURCE_ET
DELIMIT
DESCRIPTION
DESTINATION_ET
FORMATTER
NAME
REQUIRED
STYLES

Operations

getCustomSource(String, HttpServletRequest, StringBuffer)
getDestination(String, Vector, StringBuffer)
getElementNames(int, boolean)
getFormatStyle(UserDeliveryDestination)
getFormatter(QueryEngine, String)
getRequiredData(String, InfoServerElement, UserAgent, int)
getRequirements(String, int)
getSubForm(String, InfoServerElement, UserAgent, int)
loadElements(String, String)

Attribute Detail

CLASS

```
String CLASS = "CLASS"
```

Element Attribute

CUSTOMSOURCE_ET

```
int CUSTOMSOURCE_ET = 0
```

Element Type

DELIMIT

```
String DELIMIT = "~"
```

String Delimiter

DESCRIPTION

```
String DESCRIPTION = "DESC"
```

Element Attribute

DESTINATION_ET

```
int DESTINATION_ET = 1
```

Element Type

FORMATTER

```
String FORMATTER = "FORMATTER"
```

Element Attribute

NAME

```
String NAME = "NAME"
```

Element Attribute

REQUIRED

```
String REQUIRED = "REQUIRED"
```

Element Attribute

STYLES

```
String STYLES = "STYLES"
```

Element Attribute

Operation Detail

getCustomSource

```
public abstract QueryEngine getCustomSource(String source, HttpServletRequest request, StringBuffer
errorMsg)
```

getDestination

```
public abstract UserDeliveryDestination getDestination(String destination, Vector reqInfo,
StringBuffer errorMsg)
```

getElementNames

```
public abstract Vector getElementNames(int elementType, boolean addDesc)
```

getFormatStyle

```
public abstract String getFormatStyle(UserDeliveryDestination dest)
```

getFormatter

```
public abstract DataFormatter getFormatter(QueryEngine ds, String style)
```

getRequiredData

```
public abstract Vector getRequiredData(String destId, InfoServerElement element, UserAgent agent,
int elementType)
```

getRequirements

```
public abstract Vector getRequirements(String name, int elementType)
```

getSubForm

```
public abstract String getSubForm(String name, InfoServerElement element, UserAgent agent, int
elementType)
```

loadElements

```
public abstract void loadElements(String sourceElements, String destElements)
```

Load elements descriptions into hashtables

Package tcs.ain.infoserver.simpleweb

Class Diagrams

diagram simpleweb

Classes

class tcs.ain.infoserver.simpleweb.CustomEmailSourceGenerator
class tcs.ain.infoserver.simpleweb.CustomWebSourceFormGenerator
class tcs.ain.infoserver.simpleweb.CustomWebSourceGenerator
class tcs.ain.infoserver.simpleweb.InvalidParameterException

class tcs.ain.infoserver.simpleweb.JspBean
class tcs.ain.infoserver.simpleweb.PublicAgentListGenerator
class tcs.ain.infoserver.simpleweb.UserWorkFormGenerator
class tcs.ain.infoserver.simpleweb.UserWorkFormProcessor

Class Diagrams



Class Diagram *simpleweb*

Exhibit 1005, p. 293 of 332


CustomWebSourceGenerator
+SaveButtonId:String
returnJsp:String
+CustomWebSourceGenerator
+getFormForDataWorker:String
+getNewFormHtml:String
#getFormHtml:String
+getPageParserQueryEngineFrom
#getPageParserQueryEngine:Pag
#isValidLabel:boolean
#getConfidenceIcon:String
UserWorkFormGenerator
+WorkerFld:String
+IsRunningFld:String
#infoServerAccess:IRemoteRepo
#agentWorkers:DataWorker[]
#userData:UserData
#userAgent:UserAgent
+UserWorkFormGenerator
+getForms:String[]
#getIsRunningField:StringBuff
#getChangeOnlyField:StringBuf
#getTimerFields:StringBuffer
#getDestinationFields:StringB
#DoesSourceHaveEmailListener:
DoesSourceHaveSmsListener:boo
UserWorkFormProcessor
#infoServerAccess:IRemoteRepo
#userData:UserData
#userAgent:UserAgent
#dataWorker:DataWorker
request:HttpServletRequest
+UserWorkFormProcessor
+processForm:boolean
#applyIsRunning:void
#applyTimerFields:void
#getWorkerChangeOnlyFilter:Ch
#applyChangeOnly:void
#applyDestinationFields:void
#removeEmailSupport:void
#removeSmsSupport:void
#addEmailSupport:void
#addSmsSupport:void
Exceptio
InvalidParameterException
+InvalidParameterExcepti
CustomEmailSourceGenerator
+createEmailQueryEngine:NewEm
#isValidMailHost:boolean
java.lang.Obje
CustomWebSourceFormGenerator
+SaveButtonId:java.lang.Strin
returnJsp:java.lang.String
+CustomWebSourceFormGenerator
+getPageParserQueryEngineFrom
#getFormHtml:java.lang.String
+getFormForDataWorker:java.la
+getNewFormHtml:java.lang.Str
getPageParserQueryEngine:tcs.
java.lang.Obje
PublicAgentListGenerator
#infoServerAccess:tcs.ain.inf
#agentWorkers:tcs.ain.infoser
#userData:tcs.ain.appdata.Use
#userAgent:tcs.ain.infoserver
+PublicAgentListGenerator
forms:java.lang.String[]
JspBean
UserAgentId:String
UserDataId:String
+ChangeWorkerButtonId:String
+DeleteWorkerButtonId:String
+CreatePublicWorkerButtonId:S
#infoServerAccess:IRemoteRepo
#userDb:SMSAppDB
#jdbcConn:Connection
#dbCheckSql:String
+JspBean
+logout:void
+login:boolean
+isSystemAvailable:boolean
+isValidSession:boolean
+getUserData:UserData
+processCustomWorkerCreateReq
+processPublicWorkerCreateReq
+getCreatePublicWorkerForm:St
+getWorkerCount:int
+processWorkerChangeRequest:b
+getAgentWorkerEditForms:Stri
#getUserAgent:UserAgent
#validateRMIConnection:boolea
#makeRMIConnection:void
#makeDbConnection:void
#validateDbConnection:void
+ShouldWorkerQueryAlways:bool
+main:void
systemAvailable:boolean

Class Nodes

tcs.ain.infoserver.simpleweb.CustomEmailSourceGenerator
tcs.ain.infoserver.simpleweb.CustomWebSourceFormGenerator
tcs.ain.infoserver.simpleweb.CustomWebSourceGenerator
tcs.ain.infoserver.simpleweb.InvalidParameterException
tcs.ain.infoserver.simpleweb.JspBean
tcs.ain.infoserver.simpleweb.PublicAgentListGenerator
tcs.ain.infoserver.simpleweb.UserWorkFormGenerator
tcs.ain.infoserver.simpleweb.UserWorkFormProcessor

Class Node Detail

Class *tcs.ain.infoserver.simpleweb.CustomEmailSourceGenerator*

This class is used to generated a customer Email Source

Class *tcs.ain.infoserver.simpleweb.CustomWebSourceFormGenerator*

Extends:
java.lang.Object

Class *tcs.ain.infoserver.simpleweb.CustomWebSourceGenerator*

This presents all of the Public Data Sources and provides relevant information for any UserAgents that reference the public sources

Class *tcs.ain.infoserver.simpleweb.InvalidParameterException*

This can be thrown whenever a JSP passes parameters to an object and the parameters are missing, invalid, etc.

Extends:
Exception

Class *tcs.ain.infoserver.simpleweb.JspBean*

This provides some methods to simplify access to QNet and also return some dynamic HTML content.

Class *tcs.ain.infoserver.simpleweb.PublicAgentListGenerator*

Extends:
java.lang.Object

Class *tcs.ain.infoserver.simpleweb.UserWorkFormGenerator*

This presents all of the Public Data Sources and provides relevant information for any UserAgents that reference the public sources TO DO: Remove hard-coded English in the form use more constants for field ids

Class *tcs.ain.infoserver.simpleweb.UserWorkFormProcessor*

This guy applies change requests for a worker against the useragent, which is obtained by querying the session

Class Detail

Class *tcs.ain.infoserver.simpleweb.CustomEmailSourceGenerator*

```
public class CustomEmailSourceGenerator
```

This class is used to generated a customer Email Source

Operations

createEmailQueryEngine(String, String, String)
isValidMailHost(String)

Operation Detail

createEmailQueryEngine

```
public NewEmailQueryEngine createEmailQueryEngine(String hostName, String userId, String password)
```

Based on the parameters received from the request, attempt to create a NewEmailQueryEngine object. Throw an Exception with description of the problem if there is one. TO DO: DNS Validation, as with servlet, Remove Hard-Coded names

Throws:

InvalidParameterException
java.net.UnknownHostException

isValidMailHost

```
protected boolean isValidMailHost(String hostname)
```

Do the DNS MX lookup to determine if this host is valid or not

Class *tcs.ain.infoserver.simpleweb.CustomWebSourceFormGenerator*

```
public class CustomWebSourceFormGenerator
```

Extends:

java.lang.Object

Attributes

returnJsp
SaveButtonId

Constructors

CustomWebSourceFormGenerator(java.lang.String)

Operations

getFormForDataWorker(tcs.ain.infoserver.DataWorker)
getFormHtml(java.util.Hashtable, java.lang.String, java.lang.String)
getNewFormHtml(java.lang.String, java.lang.String)
getPageParserQueryEngine(tcs.ain.infoserver.DataWorker)
getPageParserQueryEngineFromForm(javax.servlet.http.HttpServletRequest)

Attribute Detail

returnJsp

```
java.lang.String returnJsp
```

SaveButtonId

```
public static java.lang.String SaveButtonId
```

Constructor Detail

CustomWebSourceFormGenerator

```
public CustomWebSourceFormGenerator(java.lang.String)
```

Operation Detail

getFormForDataWorker

```
public java.lang.String getFormForDataWorker(tcs.ain.infoserver.DataWorker)
```

getFormHtml

```
protected java.lang.String getFormHtml(java.util.Hashtable, java.lang.String, java.lang.String)
```

getNewFormHtml

```
public java.lang.String getNewFormHtml(java.lang.String, java.lang.String)
```

getPageParserQueryEngine

```
static tcs.ain.infoserver.PageParserQueryEngine
getPageParserQueryEngine(tcs.ain.infoserver.DataWorker)
```

getPageParserQueryEngineFromForm

```
public tcs.ain.infoserver.PageParserQueryEngine
getPageParserQueryEngineFromForm(javax.servlet.http.HttpServletRequest)
```

Class *tcs.ain.infoserver.simpleweb.CustomWebSourceGenerator*

```
public class CustomWebSourceGenerator
```

This presents all of the Public Data Sources and provides relevant information for any UserAgents that reference the public sources

Attributes

returnJsp
SaveButtonId

Constructors

CustomWebSourceGenerator(String)

Operations

getConfidenceIcon(int)
getFormForDataWorker(DataWorker, String)
getFormHtml(Hashtable, String, String, String)
getNewFormHtml(String, String, String)
getPageParserQueryEngine(DataWorker)
getPageParserQueryEngineFromForm(HttpServletRequest)
isValidLabel(String)

Attribute Detail

returnJsp

```
String returnJsp
```

SaveButtonId

```
public static String SaveButtonId = "save"
```

Constructor Detail

CustomWebSourceGenerator

```
public CustomWebSourceGenerator(String returnJsp)
```

Called from the JspBean when user wants forms representing workers

Operation Detail

getConfidenceIcon

```
protected String getConfidenceIcon(int level)
```

Return a representation of the confidence, based on the number.

getFormForDataWorker

```
public String getFormForDataWorker(DataWorker worker, String tableHeader)
```

Show the data selection form, but based on the PageParserQE already contained by this worker This is useful when/if we want to edit an existing Web QueryEngine

Throws:
Exception

getFormHtml

```
protected String getFormHtml(Hashtable currentSelectedItems, String href, String startText, String tableHeader)
```

Generate the form for selecting data from the href. setup the field to checked if currentSelectedItems contains the field already throw an exception if the site can not be accessed TO DO: It would be really great to sort entries by confidence level!

Throws:
Exception

getNewFormHtml

```
public String getNewFormHtml(String href, String startText, String tableHeader)
```

show a new form for selecting content startText can be null or 0 length if not relevant includes a 'saveform' submit button at the end.

Throws:
Exception

getPageParserQueryEngine

```
protected PageParserQueryEngine getPageParserQueryEngine(DataWorker worker)
```

Return a PageParserQueryEngine if associated with this data source. return null if none found

getPageParserQueryEngineFromForm

```
public PageParserQueryEngine getPageParserQueryEngineFromForm(HttpServletRequest request)
```

Create and return a PageParserQueryEngine based on the form variables.

Throws:
InvalidParameterException
java.net.UnknownHostException

isValidLabel

```
protected boolean isValidLabel(String label)
```

Does this consist of a-z or A-Z?

Class *tcs.ain.infoserver.simpleweb.InvalidParameterException*

```
public class InvalidParameterException
```

Extends:
java.lang.Exception

This can be thrown whenever a JSP passes parameters to an object and the parameters are missing, invalid, etc.

Constructors

InvalidParameterException(String)

Constructor Detail

InvalidParameterException

```
public InvalidParameterException(String s)
```

Class *tcs.ain.infoserver.simpleweb.JspBean*

```
public class JspBean
```

This provides some methods to simplify access to QNet and also return some dynamic HTML content.

Attributes

ChangeWorkerButtonId
CreatePublicWorkerButtonId
dbCheckSql
DeleteWorkerButtonId
infoServerAccess
jdbcConn
UserAgentId
UserDataId
userDb

Constructors

JspBean()

Operations

getAgentWorkerEditForms(HttpSession, String)
getCreatePublicWorkerForm(String, String)
getUserAgent(HttpSession)
getUserData(HttpSession)
getWorkerCount(HttpSession)
isSystemAvailable()
isValidSession(HttpSession)
login(String, String, HttpSession)
logout(HttpSession)
main(String[])
makeDbConnection()
makeRMIConnection()
processCustomWorkerCreateRequest(HttpSession, HttpServletRequest, QueryEngine)
processPublicWorkerCreateRequest(HttpSession, HttpServletRequest)
processWorkerChangeRequest(HttpSession, HttpServletRequest)
ShouldWorkerQueryAlways(String)
validateDbConnection()
validateRMIConnection()

Attribute Detail

ChangeWorkerButtonId

```
public static final String ChangeWorkerButtonId = "change"
```

CreatePublicWorkerButtonId

```
public static final String CreatePublicWorkerButtonId = "create"
```

dbCheckSql

```
protected String dbCheckSql = "select 'x' from dual"
```

DeleteWorkerButtonId

```
public static final String DeleteWorkerButtonId = "delete"
```

infoServerAccess

```
protected IRemoteRepository infoServerAccess = null
```

jdbcConn

```
protected Connection jdbcConn = null
```

UserAgentId

```
static final String UserAgentId = "UserAgent"
```

UserDataId

```
static final String UserDataId = "UserData"
```

userDb

```
protected SMSAppDB userDb = null
```

Constructor Detail

JspBean

```
public JspBean()
```

Operation Detail

getAgentWorkerEditForms

```
public String[] getAgentWorkerEditForms(HttpSession sess, String actionAddr)
```

Return array of HTML forms that presents all of the data workers for the user. The forms are populated with values assigned to the worker, and allow the values to be updated. Also includes forms for disabling, enabling, and remove public agents. returns null if none found

Parameter doc:

session,response, and the address of the 'page' that processes the form

getCreatePublicWorkerForm

```
public String getCreatePublicWorkerForm(String callbackPage, String buttonLabel)
```

Get the drop-down public list Form that is used to create a new Public Data Worker this is performed in here because of the complexity of the queries

getUserAgent

```
protected UserAgent getUserAgent(HttpSession session)
```

Get the UserAgent object for this session could return null if not defined protected for now; if JSP Really needs access, we can open it up

getUserData

```
public UserData getUserData(HttpSession session)
```

Get the UserData object for this session could return null if not defined

getWorkerCount

```
public int getWorkerCount(HttpSession session)
```

Return how many worker the user in this session has

isSystemAvailable

```
public boolean isSystemAvailable()
```

Validate that the RMI and DB Connections are OK. Return true if all is OK.

isValidSession

```
public boolean isValidSession(HttpSession sess)
```

Return true if this is a valid session already; i.e., has a UserAgent and UserData for the logged in user

login

```
public boolean login(String userName, String password, HttpSession sess)
```

Is this a valid username & password? If so, a session object is created and the UserAgent and UserData are loaded into the session If the session already exists for userName/password, then return true; Note that if internal connection (RMI, DB) are bad, then this returns false with no indication.

logout

```
public void logout(HttpSession sess)
```

clear out the user session.

main

```
public static void main(String[] args)
```

makeDbConnection

```
protected void makeDbConnection()
```

Make sure jdbcConn is valid. If not, report the error to out and return false

Throws:
SQLException

makeRMIConnection

```
protected void makeRMIConnection()
```

Connect the remote object to the server if necessary

Throws:
Exception

processCustomWorkerCreateRequest

```
public void processCustomWorkerCreateRequest(HttpSession sess, HttpServletRequest request,
QueryEngine qe)
```

Process the request to create a worker from the specified data source TO DO: Have this created based on admin-assignable default values

processPublicWorkerCreateRequest

```
public void processPublicWorkerCreateRequest(HttpSession sess, HttpServletRequest request)
```

Process the request to create a worker from public data source This actually just gets the public QE and then calls processCustomWorkerCreateRequest

processWorkerChangeRequest

```
public boolean processWorkerChangeRequest(HttpSession sess, HttpServletRequest request)
```

Process the worker change request that would have originated from a form generated by getAgentWorkerEditForms() return the success of the operation

ShouldWorkerQueryAlways

```
public static boolean ShouldWorkerQueryAlways(String qeId)
```

Some QueryEngines should be always queried; this determines if ID is one of them, as defined by queryAlways in the prop file

validateDbConnection

```
protected void validateDbConnection()
```

Make sure con is valid. If not, report the error to out and throw the exception

Throws:
SQLException

validateRMIConnection

```
protected boolean validateRMIConnection()
```

make connection if necessary; test the connection; if failed, try to to remake the connection return true if good connection

Class *tcs.ain.infoserver.simpleweb.PublicAgentListGenerator*

```
public class PublicAgentListGenerator
```

Extends:
java.lang.Object

Attributes
agentWorkers
infoServerAccess
userAgent
userData

Constructors
PublicAgentListGenerator(tcs.ain.infoserver.UserAgent, tcs.ain.appdata.UserData, tcs.ain.infoserver.access.IRemoteRepository)

Operations
getForms()

Attribute Detail

agentWorkers
```
protected tcs.ain.infoserver.DataWorker[] agentWorkers
```

infoServerAccess
```
protected tcs.ain.infoserver.access.IRemoteRepository infoServerAccess
```

userAgent
```
protected tcs.ain.infoserver.UserAgent userAgent
```

userData
```
protected tcs.ain.appdata.UserData userData
```

Constructor Detail

PublicAgentListGenerator
```
public PublicAgentListGenerator(tcs.ain.infoserver.UserAgent, tcs.ain.appdata.UserData,
tcs.ain.infoserver.access.IRemoteRepository)
```

Operation Detail

getForms
```
public java.lang.String[] getForms()
```

Class *tcs.ain.infoserver.simpleweb.UserWorkFormGenerator*

```
public class UserWorkFormGenerator
```

This presents all of the Public Data Sources and provides relevant information for any UserAgents that reference the public sources TO DO: Remove hard-coded English in the form use more constants for field ids

Attributes
agentWorkers
infoServerAccess
IsRunningFld
userAgent
userData
WorkerFld

Constructors
UserWorkFormGenerator(UserAgent, UserData, IRemoteRepository)

Operations

DoesSourceHaveEmailListener(IDataEventSource)
DoesSourceHaveSmsListener(IDataEventSource)
getChangeOnlyField(DataWorker)
getDestinationfields(DataWorker)
getForms(String)
getIsRunningField(DataWorker)
getTimerFields(DataWorker)

Attribute Detail

agentWorkers

```
protected DataWorker[] agentWorkers = null
```

infoServerAccess

```
protected IRemoteRepository infoServerAccess = null
```

IsRunningFld

```
public static final String IsRunningFld = "isrunning"
```

userAgent

```
protected UserAgent userAgent
```

userData

```
protected UserData userData
```

WorkerFld

```
public static final String WorkerFld = "workerid"
```

Constructor Detail

UserWorkFormGenerator

```
public UserWorkFormGenerator(UserAgent ua, UserData ud, IRemoteRepository ir)
```

Called from the JspBean when user wants forms representing workers

Operation Detail

DoesSourceHaveEmailListener

```
protected static boolean DoesSourceHaveEmailListener(IDataEventSource source)
```

Return whether this IDataEventSource has a listener of this class type (Email) Enumerates over the various listeners; recursively calls itself for Filter types

DoesSourceHaveSmsListener

```
static boolean DoesSourceHaveSmsListener(IDataEventSource source)
```

Return whether this IDataEventSource has a listener of this class type (SMS) Enumerates over the various listeners; recursively calls itself for Filter types There must be a more efficient way to do this

getChangeOnlyField

```
protected StringBuffer getChangeOnlyField(DataWorker worker)
```

return the ChangeOnly label and checkbox

getDestinationfields

```
protected StringBuffer getDestinationfields(DataWorker worker)
```

Return fields for whether SMS and/or Email Destinations are to be used

getForms

```
public String[] getForms(String actionAddr)
```

Return an array of Strings, each of which contains a form for viewing and editing the worker agents. An array is returned because the calling JSP page might want to put each form in a table cell, or something similar for formatting purposes. receive page address that will process the form action

getIsRunningField

```
protected StringBuffer getIsRunningField(DataWorker worker)
```

Get fields for whether agent is running

getTimerFields

```
protected StringBuffer getTimerFields(DataWorker worker)
```

Return the timer selection fields for the queryengine Min of hour, Hour of day, DOW, DOM

Class *tcs.ain.infoserver.simpleweb.UserWorkFormProcessor*

```
public class UserWorkFormProcessor
```

This guy applies change requests for a worker against the useragent, which is obtained by querying the session

Attributes

dataWorker
infoServerAccess
request
userAgent
userData

Constructors

UserWorkFormProcessor(UserAgent, UserData, IRemoteRepository, HttpServletRequest)

Operations

addEmailSupport(IDataSource, IDataEventSource)
addSmsSupport(IDataSource, IDataEventSource)
applyChangeOnly()
applyDestinationFields()
applyIsRunning()
applyTimerFields()
getWorkerChangeOnlyFilter()
processForm()
removeEmailSupport(IDataEventSource)
removeSmsSupport(IDataEventSource)

Attribute Detail

dataWorker

```
protected DataWorker dataWorker = null
```

infoServerAccess

```
protected IRemoteRepository infoServerAccess = null
```

request

```
HttpServletRequest request = null
```

userAgent

```
protected UserAgent userAgent = null
```

userData

```
protected UserData userData = null
```

Constructor Detail

UserWorkFormProcessor

```
public UserWorkFormProcessor(UserAgent ua, UserData ud, IRemoteRepository ir, HttpServletRequest request)
```

Called from the JspBean when we need to apply the results of having clicked on a form generated bby the UserWorkFormGenerator

Operation Detail

addEmailSupport

```
protected void addEmailSupport(IDataSource dataSource, IDataEventSource evtSource)
```

add Email listener to this source TO DO: Pull Email constants from property file

addSmsSupport

```
protected void addSmsSupport(IDataSource dataSource, IDataEventSource evtSource)
```

add Email listener to this source TO DO: Pull SMS constants from property file

applyChangeOnly

```
protected void applyChangeOnly()
```

Add or Remote a ChangeOnlyFilter, as directed by the checkbox.

Throws:
Exception

applyDestinationFields

```
protected void applyDestinationFields()
```

Apply the request to add/remove Destination TO DO: Make sure that what's happening with applyChangeOnly does not strangely affect this and vice-versa

applyIsRunning

```
protected void applyIsRunning()
```

Check and apply the isRunning field

Throws:
Exception

applyTimerFields

```
protected void applyTimerFields()
```

apply the timer fields; taking into special consideration the 'combined' time possibilities

Throws:
Exception

getWorkerChangeOnlyFilter

```
protected ChangeOnlyFilter getWorkerChangeOnlyFilter()
```

Return a ChangeOnlyFilter that may be assigned to the dataWorker. If none is assigned, return null

processForm

```
public boolean processForm()
```

Check each of the possible fields and then save the agent return false if the form could not be processed; Note: We could also throw an exception to give more detail;

removeEmailSupport

```
protected void removeEmailSupport(IDataEventSource source)
```

Remove EmailDestination from the DataEventSource

removeSmsSupport

```
protected void removeSmsSupport(IDataEventSource source)
```

Remove SmsDestination from the DataEventSource

Package *tcs.ain.infoserver.simpleweb2*zip*

Classes

class tcs.ain.infoserver.simpleweb.CustomWebSourceFormGenerator
class tcs.ain.infoserver.simpleweb.JspBean
class tcs.ain.infoserver.simpleweb.PublicAgentListGenerator
class tcs.ain.infoserver.simpleweb.UserWorkFormGenerator
class tcs.ain.infoserver.simpleweb.UserWorkFormProcessor

Class Detail

Class *tcs.ain.infoserver.simpleweb.CustomWebSourceFormGenerator*

```
public class CustomWebSourceFormGenerator
```

This presents all of the Public Data Sources and provides relevant information for any UserAgents that reference the public sources

Attributes

returnJsp
SaveButtonId

Constructors

CustomWebSourceFormGenerator(String)

Operations

getFormForDataWorker(DataWorker)
getFormHtml(Hashtable, String, String)
getNewFormHtml(String, String)
getPageParserQueryEngine(DataWorker)
getPageParserQueryEngineFromForm(HttpServletRequest)

Attribute Detail

returnJsp

```
String returnJsp
```

SaveButtonId

```
public static String SaveButtonId = "save"
```

Constructor Detail

CustomWebSourceFormGenerator

```
public CustomWebSourceFormGenerator(String returnJsp)
```

Called from the JspBean when user wants forms representing workers

Operation Detail

getFormForDataWorker

```
public String getFormForDataWorker(DataWorker worker)
```

Show the data selection form, but based on the PageParserQE already contained by this worker This is useful when/if we want to edit an existing Web QueryEngine

Throws:
Exception

getFormHtml

```
protected String getFormHtml(Hashtable currentSelectedItems, String href, String startText)
```

Generate the form for selecting data from the href. setup the field to checked if currentSelectedItems contains the field already TO DO: It would be really great to sort entries by confidence level!

Throws:
Exception

getNewFormHtml

```
public String getNewFormHtml(String href, String startText)
```

show a new form for selecting content startText can be null or 0 length if not relevant includes a 'saveform' submit button at the end.

Throws:
Exception

getPageParserQueryEngine

```
static PageParserQueryEngine getPageParserQueryEngine(DataWorker worker)
```

Return a PageParserQueryEngine if associated with this data source. return null if none found

getPageParserQueryEngineFromForm

```
public PageParserQueryEngine getPageParserQueryEngineFromForm(HttpServletRequest request)
```

Create and return a PageParserQueryEngine based on the form variables.

Throws:
Exception

Class *tcs.ain.infoserver.simpleweb.JspBean*

```
public class JspBean
```

This provides some methods to simplify access to QNet and also return some dynamic HTML content.

Attributes

infoServerAccess
jdbcConn
UserAgentId
UserDataId
userDb

Constructors

JspBean()

Operations

createEmailQueryEngine(HttpServletRequest)
getAgentWorkerEditForms(HttpSession, String)
getCreatePublicWorkerForm(String, String)
getUserAgent(HttpSession)
getUserData(HttpSession)
getWorkerCount(HttpSession)
isSystemAvailable()
isValidSession(HttpSession)
login(String, String, HttpSession)
logout(HttpSession)
main(String[])
makeDbConnection()
makeRMIConnection()

processCustomWorkerCreateRequest(HttpSession, HttpServletRequest, QueryEngine)
processPublicWorkerCreateRequest(HttpSession, HttpServletRequest)
processWorkerChangeRequest(HttpSession, HttpServletRequest)
validateDbConnection()
validateRMIConnection()

Attribute Detail

infoServerAccess

```
protected IRemoteRepository infoServerAccess = null
```

jdbcConn

```
protected Connection jdbcConn = null
```

UserAgentId

```
static final String UserAgentId = "UserAgent"
```

UserDataId

```
static final String UserDataId = "UserData"
```

userDb

```
protected SMSAppDB userDb = null
```

Constructor Detail

JspBean

```
public JspBean()
```

Operation Detail

createEmailQueryEngine

```
public NewEmailQueryEngine createEmailQueryEngine(HttpServletRequest request)
```

Based on the parameters received from the request, attempt to create a NewEmailQueryEngine object. Throw an Exception with description of the problem if there is one. TO DO: DNS Validation, as with servlet, Remove Hard-Coded names

Throws:

Exception

getAgentWorkerEditForms

```
public String[] getAgentWorkerEditForms(HttpSession sess, String actionAddr)
```

Return array of HTML forms that presents all of the data workers for the user. The forms are populated with values assigned to the worker, and allow the values to be updated. Also includes forms for disabling, enabling, and remove public agents. returns null if none found

Parameter doc:

session,response, and the address of the 'page' that processes the form

getCreatePublicWorkerForm

```
public String getCreatePublicWorkerForm(String callbackPage, String buttonLabel)
```

Get the Form that is used to create a new Public Data Worker

getUserAgent

```
public UserAgent getUserAgent(HttpSession session)
```

Get the UserAgent object for this session could return null if not defined

getUserData

```
public UserData getUserData(HttpSession session)
```

Get the UserData object for this session could return null if not defined

getWorkerCount

```
public int getWorkerCount(HttpSession session)
```

Return how many worker the user in this session has

isSystemAvailable

```
public boolean isSystemAvailable()
```

Validate that the RMI and DB Connections are OK. Return true if all is OK.

isValidSession

```
public boolean isValidSession(HttpSession sess)
```

Return true if this is a valid session already; i.e., has a UserAgent and UserData for the logged in user

login

```
public boolean login(String userName, String password, HttpSession sess)
```

Is this a valid username & password? If so, a session object is created and the UserAgent and UserData are loaded into the session If the session already exists for userName/password, then return true; Note that if internal connection (RMI, DB) are bad, then this returns false with no indication.

logout

```
public void logout(HttpSession sess)
```

clear out the user session.

main

```
public static void main(String[] args)
```

makeDbConnection

```
protected void makeDbConnection()
```

Make sure jdbcConn is valid. If not, report the error to out and return false

Throws:

SQLException

makeRMIConnection

```
protected void makeRMIConnection()
```

Connect the remote object to the server if necessary

Throws:

Exception

processCustomWorkerCreateRequest

```
public void processCustomWorkerCreateRequest(HttpSession sess, HttpServletRequest request,
QueryEngine qe)
```

Process the request to create a worker from the specified data source TO DO: Have this created based on admin-assignable default values

processPublicWorkerCreateRequest

```
public void processPublicWorkerCreateRequest(HttpSession sess, HttpServletRequest request)
```

Process the request to create a worker from public data source This actually just gets the public QE and then calls processCustomWorkerCreateRequest

processWorkerChangeRequest

```
public boolean processWorkerChangeRequest(HttpSession sess, HttpServletRequest request)
```

Process the worker change request that would have originated from a form generated by getAgentWorkerEditForms() return the success of the operation

validateDbConnection

```
protected void validateDbConnection()
```

Make sure con is valid. If not, report the error to out and throw the exception TO DO: RECONNECT ON FAILURE

Throws:

SQLException

validateRMIConnection

```
protected boolean validateRMIConnection()
```

make connection if necessary; test the connection; if failed, try to to remake the connection return true if good connection

Class *tcs.ain.infoserver.simpleweb.PublicAgentListGenerator*

```
public class PublicAgentListGenerator
```

Extends:

java.lang.Object

Attributes

agentWorkers
infoServerAccess
userAgent
userData

Constructors

PublicAgentListGenerator(tcs.ain.infoserver.UserAgent, tcs.ain.appdata.UserData, tcs.ain.infoserver.access.IRemoteRepository)

Operations

getForms()

Attribute Detail

agentWorkers

```
protected tcs.ain.infoserver.DataWorker[] agentWorkers
```

infoServerAccess

```
protected tcs.ain.infoserver.access.IRemoteRepository infoServerAccess
```

userAgent

```
protected tcs.ain.infoserver.UserAgent userAgent
```

userData

```
protected tcs.ain.appdata.UserData userData
```

Constructor Detail

PublicAgentListGenerator

```
public PublicAgentListGenerator(tcs.ain.infoserver.UserAgent, tcs.ain.appdata.UserData,
tcs.ain.infoserver.access.IRemoteRepository)
```

Operation Detail

getForms

```
public java.lang.String[] getForms()
```

Class tcs.ain.infoserver.simpleweb.UserWorkFormGenerator

```
public class UserWorkFormGenerator
```

This presents all of the Public Data Sources and provides relevant information for any UserAgents that reference the public sources

Attributes

agentWorkers
EveryIdLbl
infoServerAccess
IsRunningFld
userAgent
userData
WorkerFld

Constructors

UserWorkFormGenerator(UserAgent, UserData, IRemoteRepository)

Operations

DoesSourceHaveEmailListener(IDataEventSource)
DoesSourceHaveSmsListener(IDataEventSource)
getChangeOnlyField(DataWorker)
getDestinationfields(DataWorker)
getForms(String)
getIsRunningField(DataWorker)
getTimerFields(DataWorker)

Attribute Detail

agentWorkers

```
protected DataWorker[] agentWorkers = null
```

EveryIdLbl

```
public static String EveryIdLbl = "Every"
```

infoServerAccess

```
protected IRemoteRepository infoServerAccess = null
```

IsRunningFld

```
public static final String IsRunningFld = "isrunning"
```

userAgent

```
protected UserAgent userAgent
```

userData

```
protected UserData userData
```

WorkerFld

```
public static final String WorkerFld = "workerid"
```

Constructor Detail

UserWorkFormGenerator

```
public UserWorkFormGenerator(UserAgent ua, UserData ud, IRemoteRepository ir)
```

Called from the JspBean when user wants forms representing workers

Operation Detail

DoesSourceHaveEmailListener

```
protected static boolean DoesSourceHaveEmailListener(IDataEventSource source)
```

Return whether this IDataEventSource has a listener of this class type (Email) Enumerates over the various listeners; recursively calls itself for Filter types

DoesSourceHaveSmsListener

```
static boolean DoesSourceHaveSmsListener(IDataEventSource source)
```

Return whether this IDataEventSource has a listener of this class type (SMS) Enumerates over the various listeners; recursively calls itself for Filter types There must be a more efficient way to do this

getChangeOnlyField

```
protected StringBuffer getChangeOnlyField(DataWorker worker)
```

return the ChangeOnly label and checkbox

getDestinationfields

```
protected StringBuffer getDestinationfields(DataWorker worker)
```

Return fields for whether SMS and/or Email Destinations are to be used

getForms

```
public String[] getForms(String actionAddr)
```

Return an array of Strings, each of which contains a form for viewing and editing the worker agents. An array is returned because the calling JSP page might want to put each form in a table cell, or something similar for formatting purposes. receive page address that will process the form action

getIsRunningField

```
protected StringBuffer getIsRunningField(DataWorker worker)
```

Get fields for whether agent is running

getTimerFields

```
protected StringBuffer getTimerFields(DataWorker worker)
```

Return the timer selection fields for the queryengine Min of hour, Hour of day, DOW, DOM

Class *tcs.ain.infoserver.simpleweb.UserWorkFormProcessor*

```
public class UserWorkFormProcessor
```

This guy applies change requests for a worker against the useragent, which is obtained by querying the session

Attributes

dataWorker
infoServerAccess
request
userAgent
userData

Constructors

UserWorkFormProcessor(UserAgent, UserData, IRemoteRepository, HttpServletRequest)

Operations

addEmailSupport(IDataSource, IDataEventSource)
addSmsSupport(IDataSource, IDataEventSource)
applyChangeOnly()
applyDestinationFields()
applyIsRunning()
applyTimerFields()
getWorkerChangeOnlyFilter()
processForm()
removeEmailSupport(IDataEventSource)
removeSmsSupport(IDataEventSource)

Attribute Detail

dataWorker

```
protected DataWorker dataWorker = null
```

infoServerAccess

```
protected IRemoteRepository infoServerAccess = null
```

request

```
HttpServletRequest request = null
```

userAgent

```
protected UserAgent userAgent = null
```

userData

```
protected UserData userData = null
```

Constructor Detail

UserWorkFormProcessor

```
public UserWorkFormProcessor(UserAgent ua, UserData ud, IRemoteRepository ir, HttpServletRequest
request)
```

Called from the JspBean when we need to apply the results of having clicked on a form generated bby the UserWorkFormGenerator

Operation Detail

addEmailSupport

```
protected void addEmailSupport(IDataSource dataSource, IDataEventSource evtSource)
```

add Email listener to this source TO DO: Pull Email constants from property file

addSmsSupport

```
protected void addSmsSupport(IDataSource dataSource, IDataEventSource evtSource)
```

add Email listener to this source TO DO: Pull SMS constants from property file

applyChangeOnly

```
protected void applyChangeOnly()
```

Add or Remote a ChangeOnlyFilter, as directed by the checkbox.

Throws:
Exception

applyDestinationFields

```
protected void applyDestinationFields()
```

Apply the request to add/remove Destination TO DO: Make sure that what's happening with applyChangeOnly does not strangely affect this and vice-versa

applyIsRunning

```
protected void applyIsRunning()
```

Check and apply the isRunning field

Throws:
Exception

applyTimerFields

```
protected void applyTimerFields()
```

apply the timer fields; taking into special consideration the 'combined' time possibilities

Throws:
Exception

getWorkerChangeOnlyFilter

```
protected ChangeOnlyFilter getWorkerChangeOnlyFilter()
```

Return a ChangeOnlyFilter that may be assigned to the dataWorker. If none is assigned, return null

processForm

```
public boolean processForm()
```

Check each of the possible fields and then save the agent return false if the form could not be processed; Note: We could also throw an exception to give more detail;

removeEmailSupport

```
protected void removeEmailSupport(IDataEventSource source)
```

Remove EmailDestination from the DataEventSource

removeSmsSupport

```
protected void removeSmsSupport(IDataEventSource source)
```

Remove SmsDestination from the DataEventSource

Package *tcs.ain.infoserver.source*

Class Diagrams

diagram source

Class Diagrams

Class Diagram *source*

Package *tcs.ain.infoserver.tests*

Class Diagrams

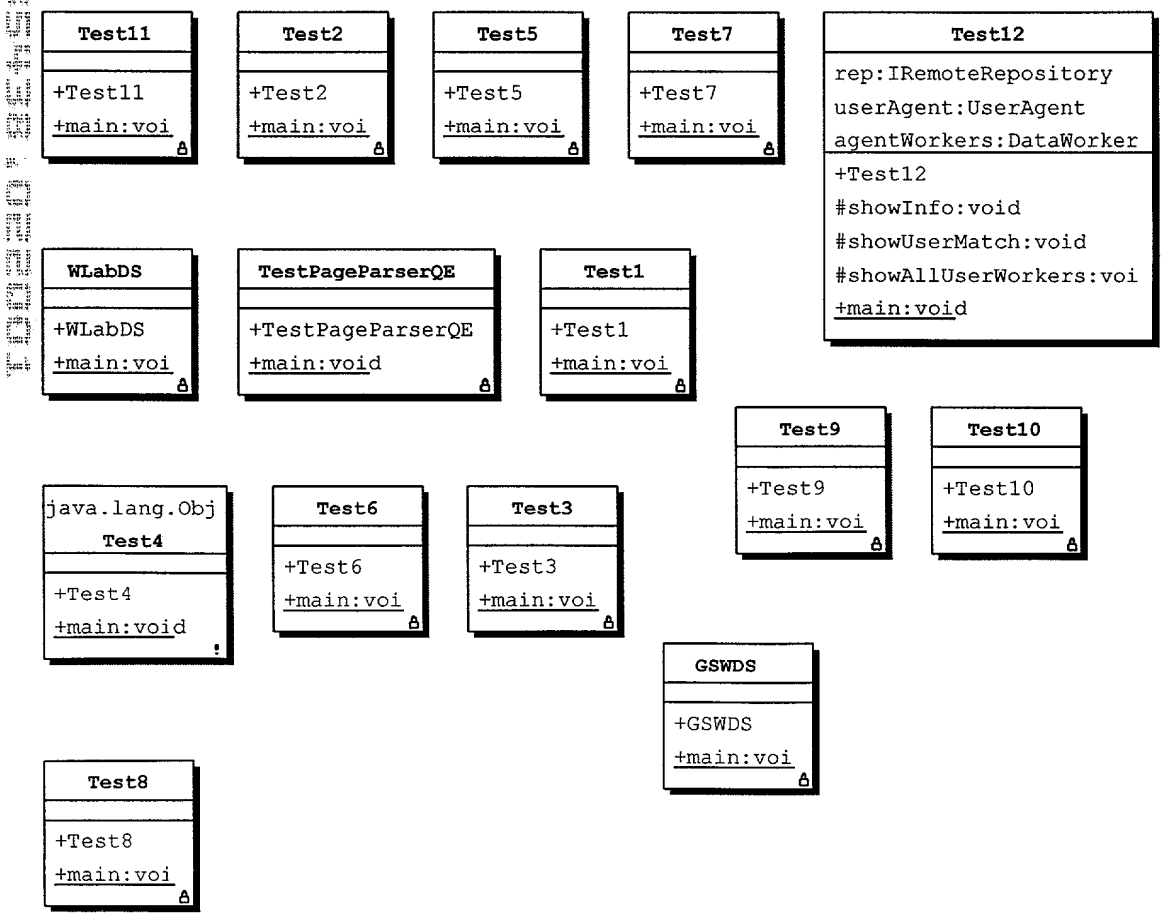
diagram tests

Classes

class tcs.ain.infoserver.tests.GSWDS
class tcs.ain.infoserver.tests.Test1
class tcs.ain.infoserver.tests.Test11
class tcs.ain.infoserver.tests.Test12
class tcs.ain.infoserver.tests.Test2
class tcs.ain.infoserver.tests.Test3
class tcs.ain.infoserver.tests.Test4
class tcs.ain.infoserver.tests.Test10
class tcs.ain.infoserver.tests.Test5
class tcs.ain.infoserver.tests.Test6
class tcs.ain.infoserver.tests.Test7
class tcs.ain.infoserver.tests.Test8
class tcs.ain.infoserver.tests.Test9
class tcs.ain.infoserver.tests.TestPageParserQE
class tcs.ain.infoserver.tests.WLabDS

Class Diagrams

Class Diagram *tests*



Class Nodes

tcs.ain.infoserver.tests.GSWDS
tcs.ain.infoserver.tests.Test1
tcs.ain.infoserver.tests.Test10
tcs.ain.infoserver.tests.Test11
tcs.ain.infoserver.tests.Test12
tcs.ain.infoserver.tests.Test2
tcs.ain.infoserver.tests.Test3
tcs.ain.infoserver.tests.Test4
tcs.ain.infoserver.tests.Test5
tcs.ain.infoserver.tests.Test6
tcs.ain.infoserver.tests.Test7
tcs.ain.infoserver.tests.Test8
tcs.ain.infoserver.tests.Test9
tcs.ain.infoserver.tests.TestPageParserQE
tcs.ain.infoserver.tests.WLabDS

Class Node Detail

Class *tcs.ain.infoserver.tests.GSWDS*

Tests the IMAP Query Engine; query every one minute

Class *tcs.ain.infoserver.tests.Test1*

This class create a WebQueryEngine, 2 DataWorkers, SimpleUserDestination and ties them all together. The dataworker will execute once, causing the XML from the horoscope web page to be seen on the screen.

Class *tcs.ain.infoserver.tests.Test10*

Adds on to Test3 by saving to the Repository.

Class *tcs.ain.infoserver.tests.Test11*

Tests the IMAP Query Engine; query every one minute

Class *tcs.ain.infoserver.tests.Test12*

This test just shows all of the Data Sources and interleaves userUser info If the user has 'signed up' with this source, then user info is shown.

Class *tcs.ain.infoserver.tests.Test2*

Adds on to Test1 by assigned an XSL doc to the DataSource, and assigning a formatter that uses that XSL doc.

Class *tcs.ain.infoserver.tests.Test3*

Adds on to Test2 by using the CalendarTimer.

Class *tcs.ain.infoserver.tests.Test4*

Extends:
java.lang.Object

Class *tcs.ain.infoserver.tests.Test5*

Loads the objects saved in Test4. Workers will start to auto-run. This is what InfoServer.main does also.

Class *tcs.ain.infoserver.tests.Test6*

Gets the InfoserverAccess object. Creates an UserAgent and saves it to the InfoServerAccess remote object.

Class *tcs.ain.infoserver.tests.Test7*

Gets the InfoserverAccess object. Retrieves the UserAgent saved in Test6.

Class *tcs.ain.infoserver.tests.Test8*

Tests the IMAP Query Engine; query every one minute

Class *tcs.ain.infoserver.tests.Test9*

Single worker, single data source for IMAP, with 2 XSL styles. Two destinations, each using a different style sheet. Dest 1 inserts the Email data into a database for the IVR reader Dest 2 displays the at the MIN.

Class *tcs.ain.infoserver.tests.TestPageParserQE*

This creates a PageParserQueryEngine and has a DataWorker query it

Class *tcs.ain.infoserver.tests.WLabDS*

Tests the IMAP Query Engine; query every one minute

Class Detail

Class *tcs.ain.infoserver.tests.GSWDS*

```
public class GSWDS
```

Tests the IMAP Query Engine; query every one minute

Constructors

GSWDS()

Operations

main(String[])

Constructor Detail

GSWDS

```
public GSWDS()
```

Throws:
Exception

Operation Detail

main

```
public static void main(String[] args)
```

Class *tcs.ain.infoserver.tests.Test1*

```
public class Test1
```

This class create a WebQueryEngine, 2 DataWorkers, SimpleUserDestination and ties them all together. The dataworker will execute once, causing the XML from the horoscope web page to be seen on the screen.

Constructors

Test1()

Operations

main(String[])

Constructor Detail

Test1

```
public Test1()
```

Operation Detail

main

```
public static void main(String[] args)
```

Class *tcs.ain.infoserver.tests.Test10*

```
public class Test10
```

Adds on to Test3 by saving to the Repository.

Constructors

Test10()

Operations

main(String[])

Constructor Detail

Test10

```
public Test10()
```

Throws:

Exception

Operation Detail

main

```
public static void main(String[] args)
```

Class *tcs.ain.infoserver.tests.Test11*

```
public class Test11
```

Tests the IMAP Query Engine; query every one minute

Constructors

Test11()

Operations

main(String[])

Constructor Detail

Test11

```
public Test11()
```

Throws:

Exception

Operation Detail

main

```
public static void main(String[] args)
```

Class *tcs.ain.infoserver.tests.Test12*

```
public class Test12
```

This test just shows all of the Data Sources and interleaves userUser info If the user has 'signed up' with this source, then user info is shown.

Attributes

agentWorkers
rep
userAgent

Constructors

Test12(String)

Operations

main(String[])
showAllUserWorkers()
showInfo()
showUserMatch(String)

Attribute Detail

agentWorkers

```
DataWorker[] agentWorkers = null
```

rep

```
IRemoteRepository rep = null
```

userAgent

```
UserAgent userAgent = null
```

Constructor Detail

Test12

```
public Test12(String uName)
```

Operation Detail

main

```
public static void main(String[] args)
```

showAllUserWorkers

```
protected void showAllUserWorkers()
```

showInfo

```
protected void showInfo()
```

Throws:

Exception

showUserMatch

```
protected void showUserMatch(String sourceKey)
```

Class *tcs.ain.infoserver.tests.Test2*

```
public class Test2
```

Adds on to Test1 by assigned an XSL doc to the DataSource, and assigning a formatter that uses that XSL doc.

Constructors

Test2()

Operations

main(String[])

Constructor Detail

Test2

```
public Test2()
```

Throws:

Exception

Operation Detail

main

```
public static void main(String[] args)
```

Class *tcs.ain.infoserver.tests.Test3*

```
public class Test3
```

Adds on to Test2 by using the CalendarTimer.

Constructors

Test3()

Operations

main(String[])

Constructor Detail

Test3

```
public Test3()
```

Throws:

Exception

Operation Detail

main

```
public static void main(String[] args)
```

Class *tcs.ain.infoserver.tests.Test4*

```
public class Test4
```

Extends:
java.lang.Object

Constructors
Test4()

Operations
main(java.lang.String[])

Constructor Detail

Test4
```
public Test4()
```

Operation Detail

main
```
public static void main(java.lang.String[])
```

Class *tcs.ain.infoserver.tests.Test5*

```
public class Test5
```

Loads the objects saved in Test4. Workers will start to auto-run. This is what InfoServer.main does also.

Constructors
Test5()

Operations
main(String[])

Constructor Detail

Test5
```
public Test5()
```

Throws:
Exception

Operation Detail

main
```
public static void main(String[] args)
```

Class *tcs.ain.infoserver.tests.Test6*

```
public class Test6
```

Gets the InfoserverAccess object. Creates an UserAgent and saves it to the InfoServerAccess remote object.

Constructors
Test6()

Operations
main(String[])

Constructor Detail

Test6

```
public Test6()
```

Throws:
Exception

Operation Detail

main

```
public static void main(String[] args)
```

Class *tcs.ain.infoserver.tests.Test7*

```
public class Test7
```

Gets the InfoserverAccess object. Retrieves the UserAgent saved in Test6.

Constructors

Test7()

Operations

main(String[])

Constructor Detail

Test7

```
public Test7()
```

Throws:
Exception

Operation Detail

main

```
public static void main(String[] args)
```

Class *tcs.ain.infoserver.tests.Test8*

```
public class Test8
```

Tests the IMAP Query Engine; query every one minute

Constructors

Test8()

Operations

main(String[])

Constructor Detail

Test8

```
public Test8()
```

Throws:
Exception

Operation Detail

main

```
public static void main(String[] args)
```

Class *tcs.ain.infoserver.tests.Test9*

```
public class Test9
```

Single worker, single data source for IMAP, with 2 XSL styles. Two destinations, each using a different style sheet. Dest 1 inserts the Email data into a database for the IVR reader Dest 2 displays the at the MIN.

Constructors

Test9(String, String, String)

Operations

main(String[])

Constructor Detail

Test9

```
public Test9(String emailServer, String emailAcct, String emailPW)
```

Throws:
Exception

Operation Detail

main

```
public static void main(String[] args)
```

Class *tcs.ain.infoserver.tests.TestPageParserQE*

```
public class TestPageParserQE
```

This creates a PageParserQueryEngine and has a DataWorker query it

Constructors

TestPageParserQE()

Operations

main(String[])

Constructor Detail

TestPageParserQE

```
public TestPageParserQE()
```

Throws:
Exception

Operation Detail

main

```
public static void main(String[] args)
```

Class *tcs.ain.infoserver.tests.WLabDS*

```
public class WLabDS
```

Tests the IMAP Query Engine; query every one minute

Constructors

WLabDS()

Operations

main(String[])

Constructor Detail

WLabDS

```
public WLabDS()
```

Throws:
Exception

Operation Detail

main

```
public static void main(String[] args)
```

Package *tcs.ain.infoserver.version1*zip*

Classes

class tcs.ain.infoserver.simpleweb.CustomWebSourceFormGenerator
class tcs.ain.infoserver.simpleweb.JspBean
class tcs.ain.infoserver.simpleweb.PublicAgentListGenerator
class tcs.ain.infoserver.simpleweb.UserWorkFormGenerator
class tcs.ain.infoserver.simpleweb.UserWorkFormProcessor

Class Detail

Class *tcs.ain.infoserver.simpleweb.CustomWebSourceFormGenerator*

```
public class CustomWebSourceFormGenerator
```

This presents all of the Public Data Sources and provides relevant information for any UserAgents that reference the public sources

Attributes

returnJsp
SaveButtonId

Constructors

CustomWebSourceFormGenerator(String)

Operations

getFormForDataWorker(DataWorker)
getFormHtml(Hashtable, String, String)
getNewFormHtml(String, String)
getPageParserQueryEngine(DataWorker)
getPageParserQueryEngineFromForm(HttpServletRequest)

Attribute Detail

returnJsp

```
String returnJsp
```

SaveButtonId

```
public static String SaveButtonId = "save"
```

Constructor Detail

CustomWebSourceFormGenerator

```
public CustomWebSourceFormGenerator(String returnJsp)
```

Called from the JspBean when user wants forms representing workers

Operation Detail

getFormForDataWorker

```
public String getFormForDataWorker(DataWorker worker)
```

Show the data selection form, but based on the PageParserQE already contained by this worker

Throws:
Exception

getFormHtml

```
protected String getFormHtml(Hashtable currentSelectedItems, String href, String startText)
```

Generate the form for selecting data from the href. setup the field to checked if currentSelectedItems contains the field already

Throws:
Exception

getNewFormHtml

```
public String getNewFormHtml(String href, String startText)
```

show a new form for selecting content startText can be null or 0 length if not relevant includes a 'saveform' submit button at the end.

Throws:
Exception

getPageParserQueryEngine

```
static PageParserQueryEngine getPageParserQueryEngine(DataWorker worker)
```

Return a PageParserQueryEngine if associated with this data source. return null if none found

getPageParserQueryEngineFromForm

```
public PageParserQueryEngine getPageParserQueryEngineFromForm(HttpServletRequest request)
```

Create and return a PageParserQueryEngine based on the form variables.

Throws:
Exception

Class *tcs.ain.infoserver.simpleweb.JspBean*

```
public class JspBean
```

This provides some methods to simplify access to QNet and also return some dynamic HTML content.

Attributes

infoServerAccess
jdbcConn
UserAgentId
UserDataId
userDb

Constructors

JspBean()

Operations

createEmailQueryEngine(HttpServletRequest)
getAgentWorkerEditForms(HttpSession, String)
getCreatePublicWorkerForm(String)
getUserAgent(HttpSession)
getUserData(HttpSession)
getWorkerCount(HttpSession)
isSystemAvailable()
isValidSession(HttpSession)
login(String, String, HttpSession)
logout(HttpSession)
main(String[])
makeDbConnection()
makeRMIConnection()
processCustomWorkerCreateRequest(HttpSession, HttpServletRequest, QueryEngine)
processPublicWorkerCreateRequest(HttpSession, HttpServletRequest)
processWorkerChangeRequest(HttpSession, HttpServletRequest)
validateDbConnection()
validateRMIConnection()

Attribute Detail

infoServerAccess

```
protected IRemoteRepository infoServerAccess = null
```

jdbcConn

```
protected Connection jdbcConn = null
```

UserAgentId

```
static final String UserAgentId = "UserAgent"
```

UserDataId

```
static final String UserDataId = "UserData"
```

userDb

```
protected SMSAppDB userDb = null
```

Constructor Detail

JspBean

```
public JspBean()
```

Operation Detail

createEmailQueryEngine

```
public NewEmailQueryEngine createEmailQueryEngine(HttpServletRequest request)
```

Based on the parameters received from the request, attempt to create a NewEmailQueryEngine object. Throw an Exception with description of the problem if there is one. TO DO: DNS Validation, as with servlet

Throws:
Exception

getAgentWorkerEditForms

```
public String[] getAgentWorkerEditForms(HttpSession sess, String actionAddr)
```

Return array of HTML forms that presents all of the data workers for the user. The forms are populated with values assigned to the worker, and allow the values to be updated. Also includes forms for disabling, enabling, and remove public agents. returns null if none found

Parameter doc:
session,response, and the address of the 'page' that processes the form

getCreatePublicWorkerForm

```
public String getCreatePublicWorkerForm(String callbackPage)
```

Get the Form that is used to create a new Public Data Worker

getUserAgent

```
public UserAgent getUserAgent(HttpSession session)
```

Get the UserAgent object for this session could return null if not defined

getUserData

```
public UserData getUserData(HttpSession session)
```

Get the UserData object for this session could return null if not defined

getWorkerCount

```
public int getWorkerCount(HttpSession session)
```

Return how many worker the user in this session has

isSystemAvailable

```
public boolean isSystemAvailable()
```

Validate that the RMI and DB Connections are OK. Return true if all is OK.

isValidSession

```
public boolean isValidSession(HttpSession sess)
```

Return true if this is a valid session already; i.e., has a UserAgent and UserData for the logged in user

login

```
public boolean login(String userName, String password, HttpSession sess)
```

Is this a valid username & password? If so, a session object is created and the UserAgent and UserData are loaded into the session If the session already exists for userName/password, then return true; Note that if internal connection (RMI, DB) are bad, then this returns false with no indication.

logout

```
public void logout(HttpSession sess)
```

clear out the user session.

main

```
public static void main(String[] args)
```

makeDbConnection

```
protected void makeDbConnection()
```

Make sure jdbcConn is valid. If not, report the error to out and return false

Throws:
SQLException

makeRMIConnection

```
protected void makeRMIConnection()
```

Connect the remote object to the server if necessary

Throws:
Exception

processCustomWorkerCreateRequest

```
public void processCustomWorkerCreateRequest(HttpSession sess, HttpServletRequest request,
QueryEngine qe)
```

Process the request to create a worker from the specified data source TO DO: Have this created based on admin-assignable default values

processPublicWorkerCreateRequest

```
public void processPublicWorkerCreateRequest(HttpSession sess, HttpServletRequest request)
```

Process the request to create a worker from public data source This actually just gets the public QE and then calls processCustomWorkerCreateRequest

processWorkerChangeRequest

```
public boolean processWorkerChangeRequest(HttpSession sess, HttpServletRequest request)
```

Process the worker change request that would have originated from a form generated by getAgentWorkerEditForms() return the success of the operation

validateDbConnection

```
protected void validateDbConnection()
```

Make sure con is valid. If not, report the error to out and throw the exception TO DO: RECONNECT ON FAILURE

Throws:
SQLException

validateRMIConnection

```
protected void validateRMIConnection()
```

make connection if necessary; test the connection; if failed, try to to remake the connection return true if good connection

Class *tcs.ain.infoserver.simpleweb.PublicAgentListGenerator*

```
public class PublicAgentListGenerator
```

Extends:
java.lang.Object

Attributes

agentWorkers
infoServerAccess
userAgent
userData

Constructors

PublicAgentListGenerator(tcs.ain.infoserver.UserAgent, tcs.ain.appdata.UserData, tcs.ain.infoserver.access.IRemoteRepository)

Operations

getForms()

Attribute Detail

agentWorkers

```
protected tcs.ain.infoserver.DataWorker[] agentWorkers
```

infoServerAccess

```
protected tcs.ain.infoserver.access.IRemoteRepository infoServerAccess
```

userAgent

```
protected tcs.ain.infoserver.UserAgent userAgent
```

userData

```
protected tcs.ain.appdata.UserData userData
```

Constructor Detail

PublicAgentListGenerator

```
public PublicAgentListGenerator(tcs.ain.infoserver.UserAgent, tcs.ain.appdata.UserData,
tcs.ain.infoserver.access.IRemoteRepository)
```

Operation Detail

getForms

```
public java.lang.String[] getForms()
```

Class *tcs.ain.infoserver.simpleweb.UserWorkFormGenerator*

```
public class UserWorkFormGenerator
```

This presents all of the Public Data Sources and provides relevant information for any UserAgents that reference the public sources

Attributes

agentWorkers
EveryIdLbl
infoServerAccess
IsRunningFld
userAgent
userData
WorkerFld

Constructors

UserWorkFormGenerator(UserAgent, UserData, IRemoteRepository)

Operations

DoesSourceHaveEmailListener(IDataEventSource)
DoesSourceHaveSmsListener(IDataEventSource)
getChangeOnlyField(DataWorker)
getDestinationfields(DataWorker)
getForms(String)
getIsRunningField(DataWorker)
getTimerFields(DataWorker)

Attribute Detail

agentWorkers

```
protected DataWorker[] agentWorkers = null
```

EveryIdLbl

```
public static String EveryIdLbl = "Every"
```

infoServerAccess

```
protected IRemoteRepository infoServerAccess = null
```

IsRunningFld

```
public static final String IsRunningFld = "isrunning"
```

userAgent

```
protected UserAgent userAgent
```

userData

```
protected UserData userData
```

WorkerFld

```
public static final String WorkerFld = "workerid"
```

Constructor Detail

UserWorkFormGenerator

```
public UserWorkFormGenerator(UserAgent ua, UserData ud, IRemoteRepository ir)
```

Called from the JspBean when user wants forms representing workers

Operation Detail

DoesSourceHaveEmailListener

```
protected static boolean DoesSourceHaveEmailListener(IDataEventSource source)
```

Return whether this IDataEventSource has a listener of this class type (Email) Enumerates over the various listeners; recursively calls itself for Filter types

DoesSourceHaveSmsListener

```
static boolean DoesSourceHaveSmsListener(IDataEventSource source)
```

Return whether this IDataEventSource has a listener of this class type (SMS) Enumerates over the various listeners; recursively calls itself for Filter types There must be a more efficient way to do this

getChangeOnlyField

```
protected StringBuffer getChangeOnlyField(DataWorker worker)
```

return the ChangeOnly label and checkbox

getDestinationfields

```
protected StringBuffer getDestinationfields(DataWorker worker)
```

Return fields for whether SMS and/or Email Destinations are to be used

getForms

```
public String[] getForms(String actionAddr)
```

Return an array of Strings, each of which contains a form for viewing and editing the worker agents. An array is returned because the calling JSP page might want to put each form in a table cell, or something similar for formatting purposes. receive page address that will process the form action

getIsRunningField

```
protected StringBuffer getIsRunningField(DataWorker worker)
```

Get fields for whether agent is running

getTimerFields

```
protected StringBuffer getTimerFields(DataWorker worker)
```

Return the timer selection fields for the queryengine Min of hour, Hour of day, DOW, DOM

Class *tcs.ain.infoserver.simpleweb.UserWorkFormProcessor*

```
public class UserWorkFormProcessor
```

This guy applies change requests for a worker against the useragent, which is obtained by querying the session

Attributes

dataWorker
infoServerAccess
request
userAgent
userData

Constructors

UserWorkFormProcessor(UserAgent, UserData, IRemoteRepository, HttpServletRequest)

Operations

addEmailSupport(IDataSource, IDataEventSource)
addSmsSupport(IDataSource, IDataEventSource)
applyChangeOnly()
applyDestinationFields()
applyIsRunning()
applyTimerFields()
getWorkerChangeOnlyFilter()
processForm()
removeEmailSupport(IDataEventSource)
removeSmsSupport(IDataEventSource)

Attribute Detail

dataWorker

```
protected DataWorker dataWorker = null
```

infoServerAccess

```
protected IRemoteRepository infoServerAccess = null
```

request

```
HttpServletRequest request = null
```

userAgent

```
protected UserAgent userAgent = null
```

userData

```
protected UserData userData = null
```

Constructor Detail

UserWorkFormProcessor

```
public UserWorkFormProcessor(UserAgent ua, UserData ud, IRemoteRepository ir, HttpServletRequest
request)
```

Called from the JspBean when we need to apply the results of having clicked on a form generated bby the UserWorkFormGenerator

Operation Detail

addEmailSupport

```
protected void addEmailSupport(IDataSource dataSource, IDataEventSource evtSource)
```

add Email listener to this source TO DO: Pull Email constants from property file

addSmsSupport

```
protected void addSmsSupport(IDataSource dataSource, IDataEventSource evtSource)
```

add Email listener to this source TO DO: Pull SMS constants from property file

applyChangeOnly

```
protected void applyChangeOnly()
```

Add or Remote a ChangeOnlyFilter, as directed by the checkbox.

Throws:
Exception

applyDestinationFields

```
protected void applyDestinationFields()
```

Apply the request to add/remove Destination TO DO: Make sure that what's happening with applyChangeOnly does not strangely affect this and vice-versa

applyIsRunning

```
protected void applyIsRunning()
```

Check and apply the isRunning field

Throws:
Exception

applyTimerFields

```
protected void applyTimerFields()
```

apply the timer fields; taking into special consideration the 'combined' time possibilities

Throws:
Exception

getWorkerChangeOnlyFilter

```
protected ChangeOnlyFilter getWorkerChangeOnlyFilter()
```

Return a ChangeOnlyFilter that may be assigned to the dataWorker. If none is assigned, return null

processForm

```
public boolean processForm()
```

Check each of the possible fields and then save the agent return false if the form could not be processed; Note: We could also throw an exception to give more detail;

removeEmailSupport

```
protected void removeEmailSupport(IDataEventSource source)
```

Remove EmailDestination from the DataEventSource

removeSmsSupport

```
protected void removeSmsSupport(IDataEventSource source)
```

Remove SmsDestination from the DataEventSource